QCPU(Q Mode)/QnACPU

MITSUBISHI

Programming Manual

(Common Instructions)

Mitsubishi Programmable
Logic Controller

MELSEC-Q

• SAFETY PRECAUTIONS •

(Always read these cautions before using the product)

Before using this product, please read this manual and the related manuals introduced in this manual, and pay full attention to safety to handle the product correctly.

Please store this manual in a safe place and make it accessible when required. Always forward a copy of the manual to the end user.

REVISIONS

* The manual number is given on the bottom left of the back cover.

Print Date	* Manual Number	Revision
Dec., 1999	SH (NA)-080039-A	First edition
Jun., 2000	SH (NA)-080039-B	Addition APPENDIX5 Correction CONTENTS, Section 3.4, 6.6.1, 6.8.6, 6.8.8, 6.8.9, 7.6.8, 9.8, 10.3, 11.2, APP 1.2, APP 4
Sep., 2000	SH (NA)-080039-C	Addition Section 9.9, 9.10, 9.11 Correction CONTENTS, Section 2.5.20, Chapter 4 Section 5.2.5, 6.6.1, 6.8.6, 7.10.1, 8.11.1, 9.3, 11.2, APP 1.2, APP 3, APP 4
Jun., 2001	SH (NA)-080039-D	Addition model Q00JCPU, Q00CPU, Q01CPU Addition Section 3.9, 11.2.1, 11.2.2, APP 1.3, APP 3.1, APP 3.2, APP 4.1, APP 4.2 Correction CONTENTS, Section 1.1, 5.3.8, 5.7.1, 6.1.5 ,6.5.2, 6.6.1, 6.8.1, 6.8.2, 6.8.4, 6.8.7, 6.8.8, 6.8.9, 7.1.2, 7.1.4, 7.1.6, 7.1.8, 7.2.1, 7.2.2, 7.2.3, 7.2.4, 7.4.2, 7.5.12, 7.6.6, 7.6.7, 7.6.9, 7.6.10, 7.7.1, 7.7.2, 7.7.3, 7.7.4, 7.9.3, 7.14.1, 9.4, 11.2.2, APP 1.2, APP 1.3, APP 2.1, APP2.1.4, APP 3.2, APP 4.2
Mar., 2002	SH (NA)-080039-E	Addition model Q12PHCPU, Q25PHCPU Addition Section 11.2.3, APP 3.3, APP 4.3 Correction CONTENTS, Section 1.1, 1.2, 3.2.2, 3.6, 3.8, 6.6.1, Chapter 9, Section 9.10, APP 1.1

Japanese Manual Version SH-080021-E

This manual confers no industrial property rights or any rights of any other kind, nor does it confer any patent licenses. Mitsubishi Electric Corporation cannot be held responsible for any problems involving industrial property rights which may occur as a result of using the contents noted in this manual.

INTRODUCTION

Thank you for purchasing the Mitsubishi MELSEC-Q Series (Q mode) and MELSEC-QnA Series of Programmable Logic Controllers.
Before using the product, please read this manual carefully to develop full familiarity with the functions and performance of the Programmable Logic Controller Q Series (Q mode)/QnA Series you have purchased, so as to ensure correct use.
A copy of this manual should be forwarded to the end User.

CONTENTS

11. ERROR CODES 11- 1 to 11 - 46

APPENDICES APP - 1 to APP - 118

Manuals

The following table lists the manuals related to the Q/QnACPU.
Please order the one you need.

Related Manuals

Manual Name	Manual Number (Model Code)
Basic model QCPU (Q mode) User's Manual (Hardware design, Maintenance and Inspection) Describes the specifications of the CPU module, power supply module, base unit, and extension cables. (Sold separately)	SH-080187 (13JR43)
Basic model QCPU (Q mode) User's Manual (Functions Explanation, Programming Fundamentals) Describes the functions, programming method, and devices to create programs with Basic model QCPU (Q mode). (Sold separately)	SH-080188 (13JR44)
High Performance model QCPU (Q mode) User's Manual (Hardware design, Maintenance and Inspection) Describes the specifications of the CPU module, power supply module, base unit, extension cables, and memory card. (Sold separately)	SH-080037 (13JL97)
High Performance model QCPU (Q mode) User's Manual (Functions Explanation, Programming Fundamentals) Describes the functions, programming method, and devices to create programs with High Performance model QCPU (Q mode). (Sold separately)	SH-080038 (13JL98)
Process CPU User’s Manual (Hardware Design, Maintenance and Inspection) Describes the specifications of the CPU module, power supply module, base unit, extension cables, and memory card. (Sold separately)	SH-080314E (13JR55)
Process CPU User’s Manual (Functions Explanation, Programming Fundamentals) Describes the functions, programming method and devices that are required to create programs. (Sold separately)	SH-080315E (13JR56)
QCPU (Q mode)/QnACPU Programming Manual (SFC) Describes the system configuration, performance specifications, functions, programming, debugging, and error codes for MELSAP3. (Sold separately)	SH-080041 (13JF60)
QCPU (Q mode) Programming Manual (MELSAP-L) Describes the system configuration, performance specifications, functions, programming, debugging, and error codes for MELSAP-L. (Sold separately)	SH-080076 (13JF61)
QCPU (Q mode)/QnACPU Programming Manual (PID Control Instructions) Describes the dedicated instructions for PID control. (Sold separately)	SH-080040 (13JF59)
QnPHCPU Programming Manual (Process Control Instructions) Describes the dedicated instructions for performing process control. (Sold separately)	SH-080316E (13JR67)
QnACPU Guidebook Aimed at people using QnACPU for the first time. Describes procedures for everything from creating programs and writing created programs to the CPU module, to debugging. Also describes how to use the QnACPU most effectively.	IB-66606 (13JF10)
Q2A(S1)/Q3A/Q4ACPU User's Manual Describes the performance, functions, and handling of the Q2ACPU(S1), Q3ACPU, and Q4ACPU, and the specifications and handling of memory cards and base units. (Sold separately)	IB-66608 (13J821)
Model Q2AS(H)CPU(S1) User's Manual Describes performance, functions, and handling of the Q2ASCPU, Q2ASCPU-S1, Q2ASHCPU, and Q2ASHCPU-S1, power supply module, memory card, specifications, and handling of the base unit. (Sold separately)	SH-3599 (13J858)

Manual Name	Manual Number (Model Code)
Q4ARCPU User's Manual Describes the Q4ARCPU features, functions, and usage. Also describes the specification and usage of the bus switching module, system management module, power supply module, memory card, and base unit. (Sold separately)	IB-66685 (13J852)
QnACPU Programming Manual (Fundamentals) Describes how to create programs, the names of devices, parameters, and types of program. (Sold separately)	IB-66614 (13JF46)
QnACPU Programming Manual (Special Function Module) Describes the dedicated instructions for special function modules available when using the Q2ACPU(S1), Q3ACPU, and Q4ACPU. (Sold separately)	SH-4013 (13JF56)
QnACPU Programming Manual (AD57 Instructions) Describes the dedicated instructions for controlling an AD57(S1) type CRT controller module available when using the Q2ACPU(S1), Q3ACPU, or Q4ACPU. (Sold separately)	IB-66617 (13JF49)
QnACPU Programming Manual (PID Control Instructions) Describes the dedicated instructions for PID control available when using the Q2ACPU(S1), Q3ACPU, or Q4ACPU. (Sold separately)	IB-66618 (13JF50)
QnACPU Programming Manual (SFC) Describes the system configuration, performance specifications, functions, programming, debugging, and error codes for MELSAP3. (Sold separately)	IB-66619 (13JF51)
For QnA/Q4AR MELSECNET/10 Network System Reference Manual Describes the general concept, specifications, and part names and settings for MELSECNET/10. (Sold separately)	IB-66690 (13JF78)
type MELSECNET, MELSECNET/B Data Link System Reference Manual Describes the general concept, specifications, and part names and settings for MELSECNET (II) and MELSECNET/B. (Sold separately)	IB-66350 (13JF70)
GX Developer Version 7 Operating Manual Describes the online functions of GX Developer Version 7 including the programming procedure, printing out procedure, monitoring procedure, and debugging procedure. (Sold separately)	SH-080166 (13JU14)
Type SW2IVD-GPPQ software package OPERATING MANUAL (Offline) Describes how to create programs and print out data when using SW2IVD-GPPQ, and the offline functions of SW2IVD-GPPQ such as file maintenance. (Included with product)	IB-66774 (13J921)
Type SW2IVD-GPPQ software package OPERATING MNUAL (Online) Describes the online functions of SW2IVD-GPPQ, including the methods for monitoring and debugging. (Included with product)	IB-66775 (13J922)
Type SW2IVD-GPPQ software package OPERATING MANUAL (SFC) Describes SFC functions such as SFC program editing and monitoring. (Included with product)	IB-66776 (13J923)

MEMO

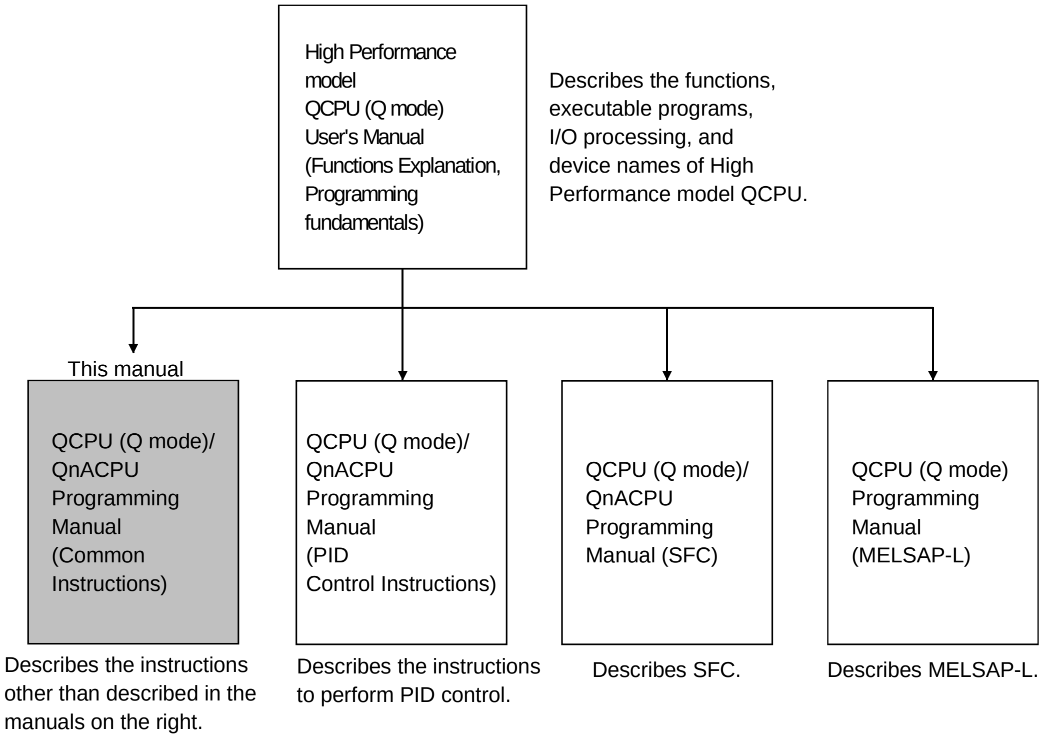
1. GENERAL DESCRIPTION

This manual describes the common instructions for QCPU, QnACPU, and Q2AS(H)CPU(S1) that are required when programming with a QCPU, QnACPU, and Q2AS(H)CPU(S1).
Common instructions are all instructions except those used for special function modules such as AJ71QC24, AJ71PT32-S3, etc.; the instructions for AD57; the instructions for PID control, and those for MELSAP3.

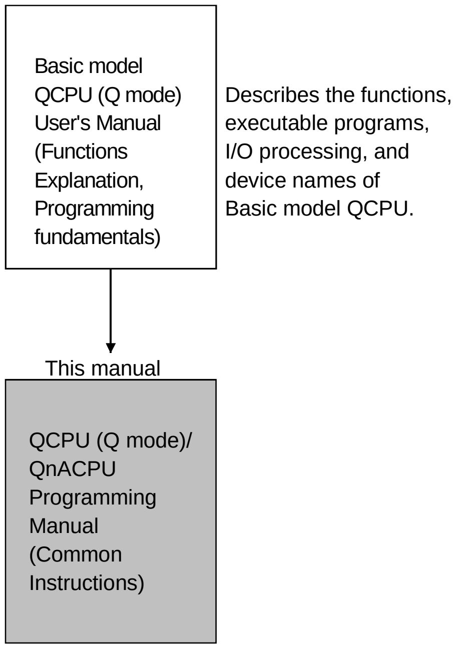
1.1 Related Programming Manuals

Before reading this manual, check the programs, I/O processes, and devices that can be used with your CPU module in the CPU Module User's Manual or in the QnACPU Programming Manual.

(1) Q02(H)CPU, Q06HCPU, Q12HCPU, Q25HCPU


High Performance model QCPU (Q mode) User's Manual (Functions Explanation, Programming fundamentals)
Describes the functions, executable programs, I/O processing, and device names of High Performance model QCPU.
This manual
QCPU (Q mode)/ QnACPU Programming Manual (Common Instructions)
Describes the instructions other than described in the manuals on the right.
QCPU (Q mode)/ QnACPU Programming Manual (PID Control Instructions)
Describes the instructions to perform PID control.
QCPU (Q mode)/ QnACPU Programming Manual (SFC)
Describes SFC.
QCPU (Q mode) Programming Manual (MELSAP-L)
Describes MELSAP-L.

(2) Q00JCPU, Q00CPU, Q01CPU



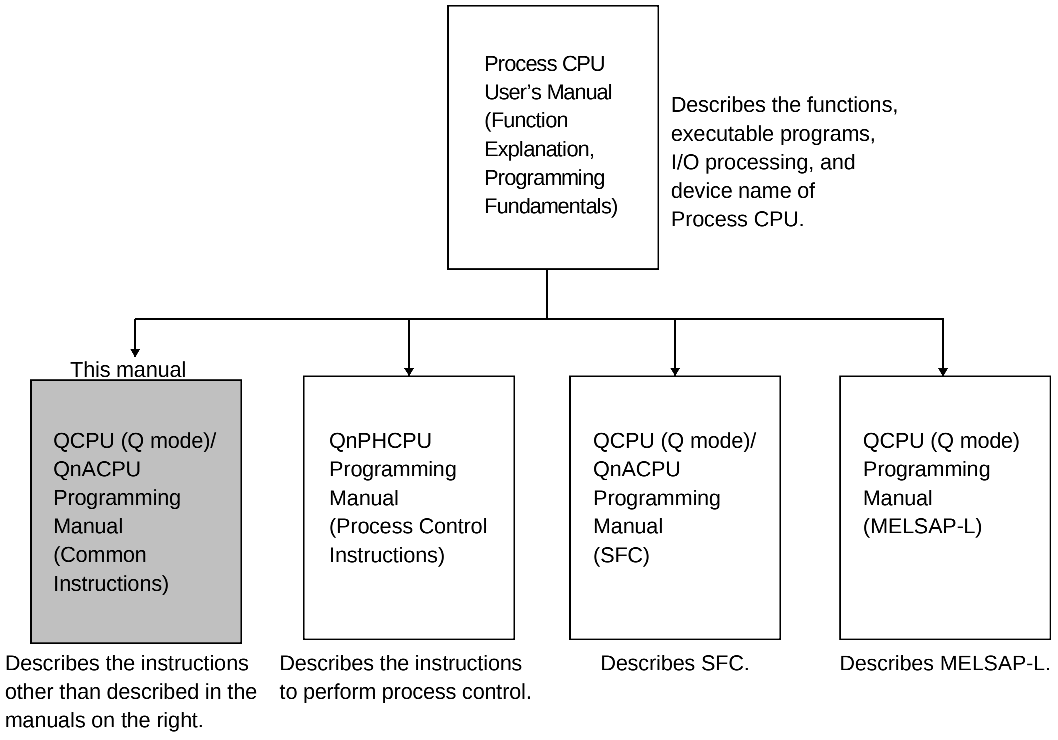
Basic model QCPU (Q mode) User's Manual (Functions Explanation, Programming fundamentals)

Describes the functions, executable programs, I/O processing, and device names of Basic model QCPU.

This manual

QCPU (Q mode)/ QnACPU Programming Manual (Common Instructions)

(3) Q12PHCPU, Q25PHCPU



Process CPU User's Manual (Function Explanation, Programming Fundamentals)

Describes the functions, executable programs, I/O processing, and device name of Process CPU.

This manual

QCPU (Q mode)/ QnACPU Programming Manual (Common Instructions)

Describes the instructions other than described in the manuals on the right.

QnPHCPU Programming Manual (Process Control Instructions)

Describes the instructions to perform process control.

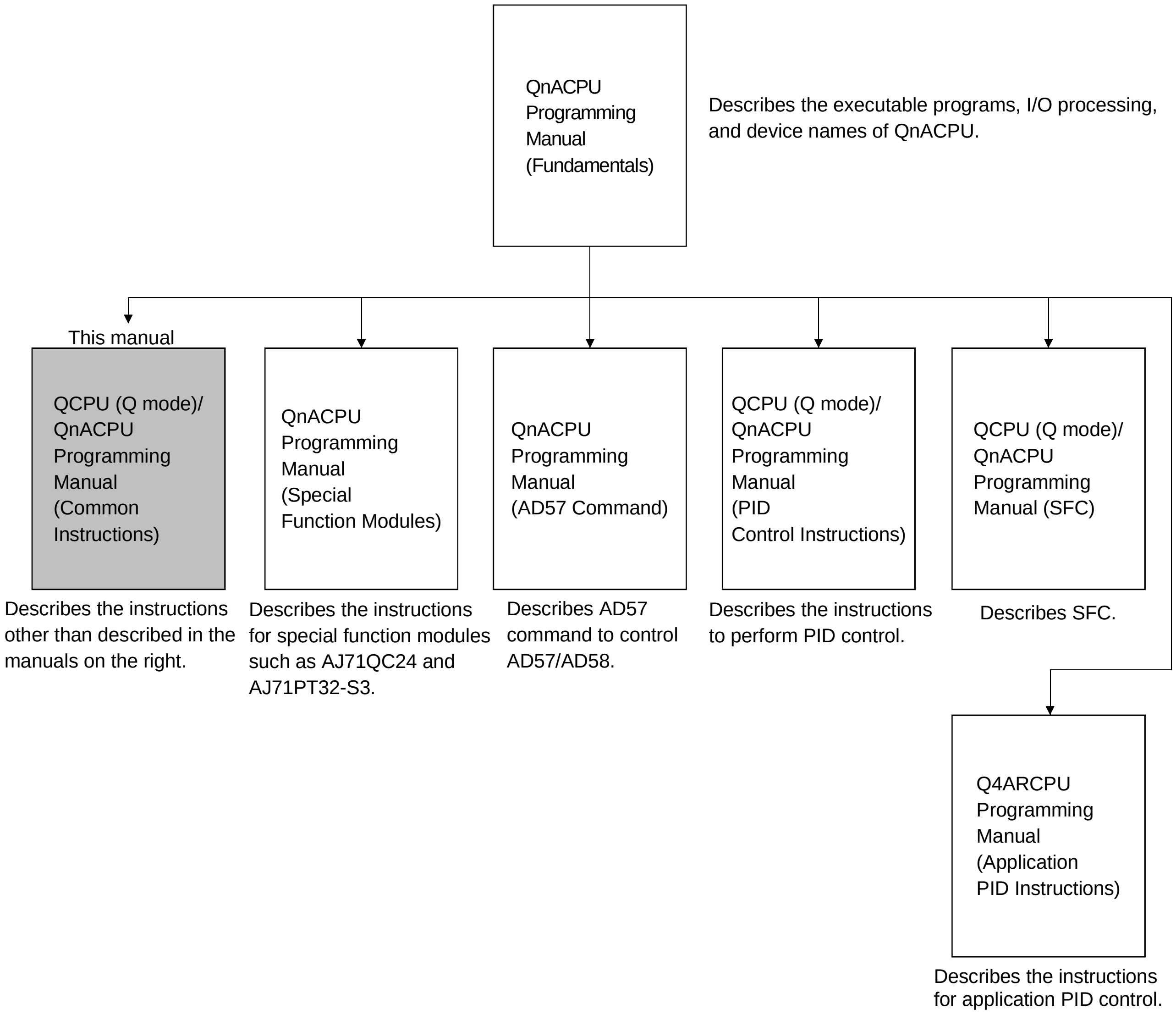
QCPU (Q mode)/ QnACPU Programming Manual (SFC)

Describes SFC.

QCPU (Q mode) Programming Manual (MELSAP-L)

Describes MELSAP-L.

(4) Q2ACPU, Q3ACPU, Q4ACPU, Q4ARCPU, Q2AS(H)CPU


QnACPU Programming Manual (Fundamentals)
This manual
QCPU (Q mode)/ QnACPU Programming Manual (Common Instructions)
QnACPU Programming Manual (Special Function Modules)
QnACPU Programming Manual (AD57 Command)
QCPU (Q mode)/ QnACPU Programming Manual (PID Control Instructions)
QCPU (Q mode)/ QnACPU Programming Manual (SFC)
Q4ARCPU Programming Manual (Application PID Instructions)

Describes the executable programs, I/O processing, and device names of QnACPU.

Describes the instructions other than described in the manuals on the right.

Describes the instructions for special function modules such as AJ71QC24 and AJ71PT32-S3.

Describes AD57 command to control AD57/AD58.

Describes the instructions to perform PID control.

Describes SFC.

Describes the instructions for application PID control.

1.2 Abbreviation and Generic Name

The module names are abbreviated as follows

Module Type Name	Abbreviation	Abbreviation in Tables	Generic Name
Q00JCPU PLC CPU Q00CPU PLC CPU Q01CPU PLC CPU Q02CPU PLC CPU Q02HCPU PLC CPU Q06HCPU PLC CPU Q12HCPU PLC CPU Q25HCPU PLC CPU Q12PHCPU PLC CPU Q25PHCPU PLC CPU	QCPU	——	CPU
Q00JCPU PLC CPU Q00CPU PLC CPU Q01CPU PLC CPU	Basic model QCPU	Basic model QCPU	
Q02CPU PLC CPU Q02HCPU PLC CPU Q06HCPU PLC CPU Q12HCPU PLC CPU Q25HCPU PLC CPU	High Performance model QCPU	High Performance model QCPU	
Q12PHCPU PLC CPU Q25PHCPU PLC CPU	Process CPU	Process CPU	
Q2ACPU(S1) PLC CPU Q3ACPU PLC CPU Q4ACPU PLC CPU	QnACPU	QnA	
Q2ASCPU(S1) PLC CPU Q2ASHCPU(S1) PLC CPU	Q2ASCPU	Q2AS	
Q4ARCPU PLC CPU	Q4ARCPU	Q4AR	
MELSECNET/H Network system (MELSECNET/H mode)	MELSECNET/H	MELSECNET/10(H)	——
MELSECNET/H Network system (MELSECNET/10 mode)			
MELSECNET/10 Network system	MELSECNET/10		
Ethernet interface module	Ethernet interface module	Ethernet interface module	——
Control and Communication Link System Master/Local Module	CC-Link module	CC-Link module	——

2. INSTRUCTION TABLES

2.1 Types of Instructions

The major types of CPU module instructions consist of sequence instructions, basic instructions, application instructions, data link instructions, QCPU instructions and redundant system instructions. These types of instructions are listed in Table 2.1 below.

Table 2.1 Types of Instructions

Types of Instructions		Meaning	Reference Chapter
Sequence instructions	Contact instruction	Operation start, series connection, parallel connection	5
	Connection instructions	Ladder block connection, creation of pulses from operation results, store/read operation results	
	Output instruction	Bit device output, pulse output, output reversal	
	Shift instruction	Bit device shift	
	Master control instruction	Master control	
	Termination instruction	Program termination	
	Other instructions	Program stop, instructions such as no operation which do not fit in the above categories	
Basic instructions	Comparison operation instruction	Comparisons such as =, >, <	6
	Arithmetic operation instruction	Addition, subtraction, multiplication or division of BIN or BCD	
	BCD ↔ BIN conversion instruction	Conversion from BCD to BIN and from BIN to BCD	
	Data transfer instruction	Transmits designated data	
	Program branch instruction	Program jumps	
	Program run control instruction	Enable or prohibit interrupt programs	
	I/O refresh	Run partial refresh	
	Other convenient instructions	Instructions for: Counter increment/decrement, teaching timer, special function timer, rotary table shortest direction control, etc.	
Application instructions	Logical operation instructions	Logical operations such as logical sum, logical product, etc.	7
	Rotation instruction	Rotation of designated data	
	Shift instruction	Shift of designated data	
	Bit processing instructions	Bit set and reset, bit test, batch reset of bit devices	
	Data processing instructions	16-bit data searches, data processing such as decoding and encoding	
	Structure creation instructions	Repeated operation, subroutine program calls, Index modification in ladder units	
	Table operation instruction	Read/Write of FIFO table	
	Buffer memory access instruction	Data read/write for special function modules	
	Display instructions	Print ASCII code, LED character display, etc.	
	Debugging and failure diagnosis instructions	Check, status check, sampling trace, program trace	
	Character string processing instructions	Conversion between BIN/BCD and ASCII; conversion between BIN and character string; conversion between floating decimal point data and character strings, character string processing, etc.	
	Special function instructions	Trigonometric functions, conversion between angles and radians, exponential operations, automatic logarithms, square roots	
	Data control instructions	Upper and lower limit controls, dead band controls, zone controls	
	Switching instructions	File register block No. switches, designation of file registers and comment files	
	Clock instructions	Read/write of year, month, day, hour, minute, second, and day of the week; conversion between time statement (hour, minute, second) and seconds	
	Peripheral device instructions	I/O to peripheral devices	
	Program instructions	Instructions to switch program execution conditions	
	Other instructions	Instructions that do not fit in the above categories, such as watchdog timer reset instructions and timing clock instructions	
Data link instructions	Link refresh instructions	Designated network refresh	8
	Instructions dedicated to QnA links	Read/write of data from other stations; data transmission signals to other stations; processing requests to other stations	
	Instructions for A-series-compatible link	Read/write for designated station word device, read/write data from remote I/O station special function module	
	Routing information read/write instructions	Reads, writes, and registers routing information.	
QCPU instructions	Instruction for QCPU	Reading module information; trace set/reset; reading/writing binary data; load/unload/load + unload program from memory card; high-speed block transfer of file register	9
Redundant system instructions	Instructions for Q4ARCPU	Operation mode setting during CPU startup; operation mode setting instructions during CPU switch; data tracking; buffer memory batch refresh	10

2.2 How to Read Instruction Tables

The instruction tables found from Section 2.3 to 2.6 have been made according to the following format:

Table 2.2 How to Read Instruction Tables

Category	Instruction Symbols	Symbol	Processing Details	Execution Condition	Number of Basic Steps	Subset	See for Description
BIN 16-bit addition and subtraction operations	+	+ S D	• (D)+(S)→(D)	⎍	3	●	6-16
	+P	+P S D		↑⎍			
	+	+ S1 S2 D	• (S1)+(S2)→(D)	⎍	4	●	6-18
	+P	+P S1 S2 D		↑			
①	②	③	④	⑤	⑥	⑦	⑧

Description

①...........Classifies instructions according to their application

②...........Indicates the instruction symbol added to the instruction in a program
Instruction code is built around the 16-bit instruction. The following notations are used to mark 32-bit instructions, instructions executed only at the leading edge of OFF to ON, real number instructions, and character string instructions:

- 32-bit instruction .... The letter "D" is added to the first line of the instruction

  Example + ➡ D+
  16-bit instruction 32-bit instruction

- Instructions executed only at the leading edge of OFF to ON .... The letter "P" is added to the end of the instruction

  Example + ➡ +P
  Instructions executed when ON
  Instructions executed only at the leading edge of OFF to ON

- Real number instructions .... The letter "E" is added to the first line of the instruction

  Example + ➡ E+
  Real number instructions

- Character string instructions ...... A dollar sign "$" is added to the first line of the instruction

  Example + ➡ $+
  Character string instructions

③...........Shows symbol diagram on the ladder

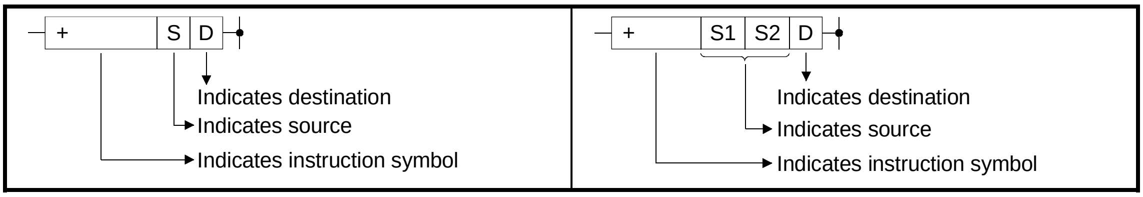


Fig. 2.1 Shows Symbol Diagram on the Ladder

Destination .... Indicates where data will be sent after operation
Source .... Stores data prior to operation

④...........Indicates the type of processing that is performed by individual instructions

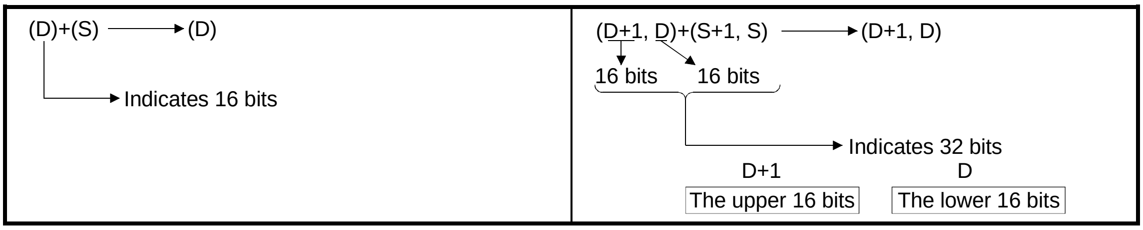
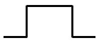
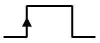
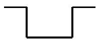
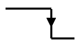


Fig. 2.2 Type of Processing Performed by Individual Instructions

⑤...........The details of conditions for the execution of individual instructions are as follows:

Symbol	Execution Condition
No symbol recorded	Instruction executed under normal circumstances, with no regard to the ON/OFF status of conditions prior to the instruction. If the preconditions is OFF, the instruction will conduct OFF processing.
	Executed during ON; instruction is executed only while the precondition is ON. If the preconditions is OFF, the instruction is not executed, and no processing is conducted.
	Executed once at ON; instruction executed only at leading edge when precondition goes from OFF to ON. Following execution, instruction will not be executed and no processing conducted even if condition remains ON.
	Executed during OFF; instruction is executed only while the precondition is OFF. If the precondition is ON, the instruction is not executed, and no processing is conducted.
	Executed once at OFF; instruction executed only at trailing edge when precondition goes from ON to OFF. Following execution, instruction will not be executed and no processing conducted even if condition remains OFF.

⑥...........Indicates the basic number of steps for individual instructions.
See Section 3.8 for a description of the number of steps.

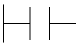
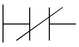
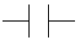
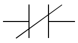
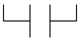
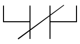
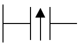
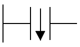
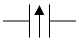
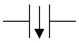
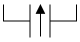
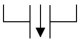
⑦...........The "●" mark indicates instructions for which subset processing is possible.
See Section 3.5 for details on subset processing.

⑧...........Indicates the page numbers where the individual instructions are explained.

2.3 Sequence Instructions

2.3.1 Contact Instruction

Table 2.3 Contact Instructions

Category	Instruction Symbols	Symbol	Processing Details	Execution Condition	Number of Basic Steps	Subset	See for Description
Contact	LD		• Starts logic operation (Starts a contact logic operation)		＊1	● ＊3	5-2
	LDI		• Starts logical NOT operation (Starts b contact logic operation)				
	AND		• Logical product (a contact series connection)				
	ANI		• Logical product NOT (b contact series connection)				
	OR		• Logical sum (a contact parallel connection)				
	ORI		• Logical sum NOT (b contact parallel connection)				
	LDP		• Starts leading edge pulse operation		＊2	● ＊3	5-5
	LDF		• Starts trailing edge pulse operation				
	ANDP		• Leading edge pulse series connection				
	ANDF		• Trailing edge pulse series connection				
	ORP		• Leading edge pulse parallel connection				
	ORF		• Trailing edge pulse parallel connection				

REMARKS

1) ＊1 : The number of steps may vary depending on the device being used.

Device	Number of Steps
Internal device, file register (R0 to R32767)	1
Direct access input (DX)	2
Devices other than above	3

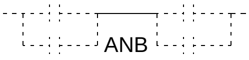
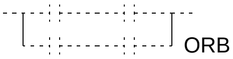
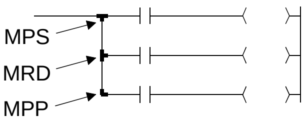
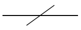
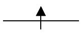
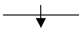
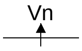
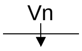
2) ＊2 : The number of steps may vary depending on the device and type of CPU module being used.

Device	Number of Steps	
	QCPU	QnACPU
Internal device, file register (R0 to R32767)	1	2
Direct access input (DX)	2	2
Devices other than above	3	3

3) ＊3: The subset is effective only with QCPU.

2.3.2 Connection instructions

Table 2.4 Connection Instructions

Category	Instruction Symbols	Symbol	Processing Details	Execution Condition	Number of Basic Steps	Subset	See for Description
Connection	ANB		• AND between logical blocks (Series connection between logical blocks)		1		5-7
	ORB		• OR between logical blocks (Series connection between logical blocks)				
	MPS		• Memory storage of operation results		1		5-9
	MRD		• Read of operation results stored with MPS instruction				
	MPP		• Read and reset of operation results stored with MPS instruction				
	INV		• Inversion of operation result		1		5-13
	MEP		• Conversion of operation result to leading edge pulse		1		5-14
	MEF		• Conversion of operation result to trailing edge pulse				
	EGP		• Conversion of operation result to leading edge pulse (Stored at Vn)		1		5-16
	EGF		• Conversion of operation result to trailing edge pulse (Stored at Vn)		＊1		

REMARKS

＊1:The number of steps may vary depending on the type of CPU module being used.

Component	Number of basic steps
High Performance model QCPU Process CPU QnACPU	1
Basic model QCPU	2

2.3.3 Output instructions

Table 2.5 Output Instructions

Category	Instruction Symbols	Symbol	Processing Details	Execution Condition	Number of Basic Steps	Subset	See for Description
Output	OUT	—( )—|	• Device output		＊1		5-18
	SET	—[SET D]—|	• Set device	＊2 ⎍ (↑)	＊1		5-28 5-32
	RST	—[RST D]—|	• Reset device	＊2 ⎍ (↑)	＊1		5-30 5-32
	PLS	—[PLS D]—|	• Generates 1 cycle program pulse at leading edge of input signal	↑	2		5-34
	PLF	—[PLF D]—|	• Generates 1 cycle program pulse at trailing edge of input signal	↓			
	FF	—[FF D]—|	• Reversal of device output	↑	2		5-36
	DELTA	—[DELTA D]—|	• Pulse conversion of direct output	⎍	2		5-38
	DELTAP	—[DELTAP D]—|		↑			

REMARKS

1) ＊1: The number of steps may vary depending on the device in use.
See description pages of individual instructions for number of steps.

2) ＊2: The ↑ execution condition applies only when an annunciator (F) is in use.

2.3.4 Shift instructions

Table 2.6 Shift Instructions

Category	Instruction Symbols	Symbol	Processing Details	Execution Condition	Number of Basic Steps	Subset	See for Description
Shift	SFT	—[SFT D]—|	• 1-bit shift of device	⎍	2		5-40
	SFTP	—[SFTP D]—|		↑			

2.3.5 Master control instructions

Table 2.7 Master Control Instructions

Category	Instruction Symbols	Symbol	Processing Details	Execution Condition	Number of Basic Steps	Subset	See for Description
Master control	MC	MC n D	• Starts master control		2		5-42
	MCR	MCR n	• Resets master control		1		

2.3.6 Termination instruction

Table 2.8 Termination Instructions

Category	Instruction Symbols	Symbol	Processing Details	Execution Condition	Number of Basic Steps	Subset	See for Description
Program end	FEND	FEND	• Termination of main program		1		5-46
	END	END	• Termination of sequence program				5-48

2.3.7 Other instructions

Table 2.9 Other Instructions

Category	Instruction Symbols	Symbol	Processing Details	Execution Condition	Number of Basic Steps	Subset	See for Description
Stop	STOP	STOP	• Terminates sequence operation after input condition has been met • Sequence program is executed by placing the RUN/STOP key switch back in the RUN position	⎍	1		5-50
Ignored	NOP	—	• Ignored (For program deletion or space)		1		5-52
	NOPLF	NOPLF	• Ignored (To change pages during printouts)				
	PAGE	PAGE n	• Ignored (Subsequent programs will be controlled from step 0 of page n)				

2.4 Basic Instructions

2.4.1 Comparison operation instruction

Table 2.10 Comparison Operation Instruction

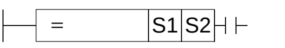
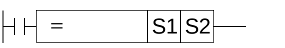
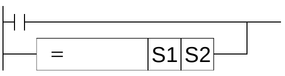
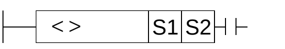
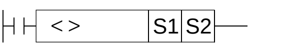
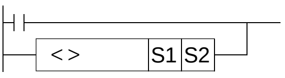
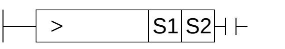
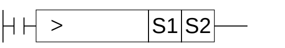
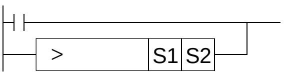
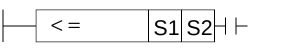
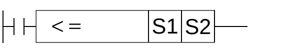
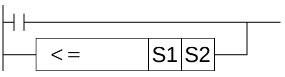
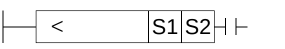
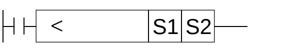
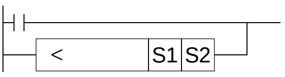
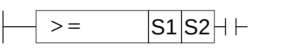
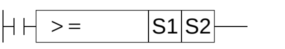
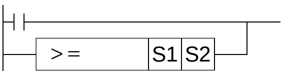
Category	Instruction Symbols	Symbol	Processing Details	Execution Condition	Number of Basic Steps	Subset	See for Description
16-bit data compari-sons	LD=		• Conductive status when (S1) = (S2) • Non-conductive status when (S1) ≠ (S2)		3	●	6-2
	AND=						
	OR=						
	LD<>		• Conductive status when (S1) ≠ (S2) • Non-conductive status when (S1) = (S2)		3	●	6-2
	AND<>						
	OR<>						
	LD>		• Conductive status when (S1) > (S2) • Non-conductive status when (S1) ≤ (S2)		3	●	6-2
	AND>						
	OR>						
	LD<=		• Conductive status when (S1) ≤ (S2) • Non-conductive status when (S1) > (S2)		3	●	6-2
	AND<=						
	OR<=						
	LD<		• Conductive status when (S1) < (S2) • Non-conductive status when (S1) ≥ (S2)		3	●	6-2
	AND<						
	OR<						
	LD>=		• Conductive status when (S1) ≥ (S2) • Non-conductive status when (S1) < (S2)		3	●	6-2
	AND>=						
	OR>=						

Table 2.10 Comparison Operation Instructions (Continued)

Category	Instruction Symbols	Symbol	Processing Details	Execution Condition	Number of Basic Steps	Subset	See for Description
32-bit data compari-sons	LDD=	D= S1 S2	• Conductive status when (S1+1, S1) = (S2+1, S2) • Non-Conductive status when (S1+1, S1) ≠ (S2+1, S2)		∗1	●	6-4
	ANDD=	D= S1 S2					
	ORD=	D= S1 S2					
	LDD<>	D< > S1 S2	• Conductive status when (S1+1, S1) ≠ (S2+1, S2) • Non-Conductive status when (S1+1, S1) = (S2+1,S2)		∗1	●	6-4
	ANDD<>	D< > S1 S2					
	ORD<>	D< > S1 S2					
	LDD>	D> S1 S2	• Conductive status when (S1+1, S1) > (S2+1, S2) • Non-Conductive status when (S1+1, S1) ≤ (S2+1, S2)		∗1	●	6-4
	ANDD>	D > S1 S2					
	ORD>	D> S1 S2					
	LDD<=	D< = S1 S2	• Conductive status when (S1+1, S1) ≤ (S2+1, S2) • Non-Conductive status when (S1+1, S1) > (S2+1, S2)		∗1	●	6-4
	ANDD<=	D< = S1 S2					
	ORD<=	D< = S1 S2					
	LDD<	D< S1 S2	• Conductive status when (S1+1, S1) <(S2+1, S2) • Non-Conductive status when (S1+1, S1) ≥ (S2+1, S2)		∗1	●	6-4
	ANDD<	D< S1 S2					
	ORD<	D< S1 S2					
	LDD>=	D> = S1 S2	• Conductive status when (S1+1, S1) ≥ (S2+1, S2) • Non-Conductive status when (S1+1, S1) < (S2+1, S2)		∗1	●	6-4
	ANDD>=	D> = S1 S2					
	ORD>=	D> = S1 S2					

REMARK

∗1 : The number of steps may vary depending on the device and type of CPU module being used.

Component	Nomber of basic steps	
High Performance model QCPU Process CPU	(1) When using the following devices only • Word device : Internal device (except for file register ZR) • Bit device : Devices whose device Nos. are multiples of 16, whose digit designation is K8, and which use no index modification. • Constant : No limitations	5
	(2) When using devices other than (1)	3
Basic model QCPU QnCPU	3	

Note 1:With High Performance module QCPU, (1) requires more number of steps, while it can process the steps faster, as compared with (2).

Note 2:The number of steps may increase due to the conditions described in Section 3.8.

Table 2.10 Comparison Operation Instructions (Continued)

Category	Instruction Symbols	Symbol	Processing Details	Execution Condition	Number of Basic Steps	Subset	See for Description
Real number data comparisons	LDE=	E = S1 S2	• Conductive status when (S1+1, S1) = (S2+1, S2) • Non-Conductive status when (S1+1, S1) ≠ (S2+1, S2)		3		6-6
	ANDE=	E = S1 S2					
	ORE=	E = S1 S2					
	LDE<>	E < > S1 S2	• Conductive status when (S1+1, S1) ≠ (S2+1, S2) • Non-Conductive status when (S1+1, S1) = (S2+1, S2)		3		6-6
	ANDE<>	E < > S1 S2					
	ORE<>	E < > S1 S2					
	LDE>	E > S1 S2	• Conductive status when (S1+1, S1) > (S2+1, S2) • Non-Conductive status when (S1+1, S1) ≤ (S2+1, S2)		3		6-6
	ANDE>	E> S1 S2					
	ORE>	E > S1 S2					
	LDE<=	E< = S1 S2	• Conductive status when (S1+1, S1) ≤ (S2+1, S2) • Non-Conductive status when (S1+1, S1) > (S2+1, S2)		3		6-6
	ANDE<=	E< = S1 S2					
	ORE<=	E < = S1 S2					
	LDE<	E < S1 S2	• Conductive status when (S1+1, S1) < (S2+1, S2) • Non-Conductive status when (S1+1, S1) ≥ (S2+1, S2)		3		6-6
	ANDE<	E < S1 S2					
	ORE<	E < S1 S2					
	LDE>=	E > = S1 S2	• Conductive status when (S1+1, S1) ≥ (S2+1, S2) • Non-Conductive status when (S1+1, S1) < (S2+1, S2)		3		6-6
	ANDE>=	E> = S1 S2					
	ORE>=	E > = S1 S2					

Table 2.10 Comparison Operation Instructions (Continued)

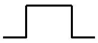
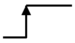
Category	Instruction Symbols	Symbol	Processing Details	Execution Condition	Number of Basic Steps	Subset	See for Description
Character string data compari-sons	LD$=	$ = S1 S2	• Compares character string S1 and character string S2 one character at a time. ∗ • Conductive status when (character string S1) = (character string S2) • Non-Conductive status when (character string S1) ≠ (character string S2)		3		6-8
	AND$=	$ = S1 S2					
	OR$=	$ = S1 S2					
	LD$<>	$ < > S1 S2	• Compares character string S1 and character string S2 one character at a time. ∗ • Conductive status when (character string S1) ≠ (character string S2) • Non-Conductive status when (character string S1) = (character string S2)		3		6-8
	AND$<>	$ < > S1 S2					
	OR$<>	$ < > S1 S2					
	LD$>	$ > S1 S2	• Compares character string S1 and character string S2 one character at a time. ∗ • Conductive status when (character string S1) > (character string S2) • Non-Conductive status when (character string S1) ≤ (character string S2)		3		6-8
	AND$>	$ > S1 S2					
	OR$>	$ > S1 S2					
	LD$<=	$ < = S1 S2	• Compares character string S1 and character string S2 one character at a time.∗ • Conductive status when (character string S1) ≤ (character string S2) • Non-Conductive status when (character string S1) > (character string S2)		3		6-8
	AND$<=	$ < = S1 S2					
	OR$<=	$ < = S1 S2					
	LD$<	$ < S1 S2	• Compares character string S1 and character string S2 one character at a time. ∗ • Conductive status when (character string S1) < (character string S2) • Non-Conductive status when (character string S1) ≥ (character string S2)		3		6-8
	AND$<	$ < S1 S2					
	OR$<	$ < S1 S2					
	LD$>=	$ > = S1 S2	• Compares character string S1 and character string S2 one character at a time. ∗ • Conductive status when (character string S1) ≥ (character string S2) • Non-Conductive status when (character string S1) < (character string S2)		3		6-8
	AND$>=	$ > = S1 S2					
	OR$>=	$ > = S1 S2					

REMARK

1) ∗ : The conditions under which character string comparisons can be made are as shown below

• Match:	All characters in the strings must match
• Larger string:	If character strings are different, determines the string with the largest number of character codes If the lengths of the character strings are different, determines the longest character string
• Smaller string:	If the character strings are different, determines the string with the smallest number of character codes If the lengths of the character strings are different, determines the shortest character string

Table 2.10 Comparison Operation Instructions (Continued)

Category	Instruction Symbols	Symbol	Processing Details	Execution Condition	Number of Basic Steps	Subset	See for Description
Block data compari-sons	BKCMP=	BKCMP = | S1 | S2 | D | n	• Compares n points of data from S1 with n points of data from S2 in 1-word units, and stores the results of the comparison at n points from the bit device designated by (D).		5		6-12
	BKCMP<>	BKCMP< > | S1 | S2 | D | n					
	BKCMP>	BKCMP > | S1 | S2 | D | n					
	BKCMP<=	BKCMP< = | S1 | S2 | D | n					
	BKCMP<	BKCMP < | S1 | S2 | D | n					
	BKCMP>=	BKCMP> = | S1 | S2 | D | n					
	BKCMP=P	BKCMP=P | S1 | S2 | D | n					
	BKCMP< >P	BKCMP< >P | S1 | S2 | D | n					
	BKCMP>P	BKCMP > P | S1 | S2 | D | n					
	BKCMP<=P	BKCMP< =P | S1 | S2 | D | n					
	BKCMP<P	BKCMP < P | S1 | S2 | D | n					
	BKCMP>=P	BKCMP> =P | S1 | S2 | D | n					

2.4.2 Arithmetic operation instructions

Table 2.11 Arithmetic Operation Instructions

Category	Instruction Symbols	Symbol	Processing Details	Execution Condition	Number of Basic Steps	Subset	See for Description
BIN 16-bit addition and subtraction operations	+	+ S D	• (D)+(S) → (D)	⎍	3	●	6-16
	+P	+P S D		↑			
	+	+ S1 S2 D	• (S1)+(S2) → (D)	⎍	4	●	6-18
	+P	+P S1 S2 D		↑			
	-	− S D	• (D) - (S) → (D)	⎍	3	●	6-16
	-P	−P S D		↑			
	-	− S1 S2 D	• (S1) - (S2) → (D)	⎍	4	●	6-18
	-P	−P S1 S2 D		↑			
BIN 32-bit addition and subtraction operations	D+	D+ S D	• (D+1, D)+(S+1, S) → (D+1, D)	⎍	∗1	●	6-20
	D+P	D+P S D		↑			
	D+	D+ S1 S2 D	• (S1+1, S1)+(S2+1, S2) → (D+1, D)	⎍	∗2	●	6-22
	D+P	D+P S1 S2 D		↑			
	D-	D− S D	• (D+1, D)-(S+1, S) → (D+1, D)	⎍	∗1	●	6-20
	D-P	D−P S D		↑			
	D-	D− S1 S2 D	• (S1+1, S1)-(S2+1, S2) → (D+1, D)	⎍	∗2	●	6-22
	D-P	D−P S1 S2 D		↑			
BIN 16-bit multiplication and division operations	∗	∗ S1 S2 D	• (S1) ∗ (S2) → (D+1, D)	⎍	∗3	●	6-24
	∗P	∗P S1 S2 D		↑			
	/	/ S1 S2 D	• (S1)/(S2) → Quotient(D), Remainder (D+1)	⎍	4	●	6-24
	/P	/P S1 S2 D		↑			
BIN 32-bit multiplication and division operations	D∗	D∗ S1 S2 D	• (S1+1, S1) ∗ (S2+1, S2) → (D+3, D+2, D+1, D)	⎍	4	●	6-26
	D∗P	D∗P S1 S2 D		↑			
	D/	D/ S1 S2 D	• (S1+1, S1)/(S2+1, S2) → Quotient (D+1, D), Remainder (D+3, D+2)	⎍	4	●	6-26
	D/P	D/P S1 S2 D		↑			

REMARKS

1) ＊1:The number of steps may vary depending on the device and type of CPU module being used.

Component	Nomber of basic steps	
High Performance model QCPU Process CPU	(1) When using the following devices only • Word device : Internal device (except for file register ZR) • Bit device : Devices whose device Nos. are multiples of 16, whose digit designation is K8, and which use no index modification. • Constant : No limitations	Note 1) 5
	(2) When using devices other than (1)	Note 2) 3
Basic model QCPU QnCPU	3	Note 2)

Note 1:With High Performance module QCPU, (1) requires more number of steps, while it can process the steps faster, as compared with (2).

Note 2:The number of steps may increase due to the conditions described in Section 3.8.

2) ＊2:The number of steps may vary depending on the device and type of CPU module being used.

Component	Nomber of basic steps	
High Performance model QCPU Process CPU	(1) When using the following devices only • Word device : Internal device (except for file register ZR) • Bit device : Devices whose device Nos. are multiples of 16, whose digit designation is K8, and which use no index modification. • Constant : No limitations	Note 1) 6
	(2) When using devices other than (1)	Note 2) 4
Basic model QCPU QnCPU	4	Note 2)

Note 1:With High Performance module QCPU, (1) requires more number of steps, while it can process the steps faster, as compared with (2).

Note 2:The number of steps may increase due to the conditions described in Section 3.8.

3) ＊3:The number of steps may vary depending on the device and type of CPU module being used.

Component	Nomber of basic steps	
QCPU	(1) When using the following devices only • Word device : Internal device (except for file register ZR) • Bit device : Devices whose device Nos. are multiples of 16, whose digit designation is K8, and which use no index modification. • Constant : No limitations	3
	(2) When using devices other than (1)	4
QnCPU	4	

Table 2.11 Arithmetic Operation Instructions (Continued)

Category	Instruction Symbols	Symbol	Processing Details	Execution Condition	Number of Basic Steps	Subset	See for Description
BCD 4-digit addition and subtraction operations	B+	[B+ S D]	• (D)+(S) → (D)	⎍	3	●	6-28
	B+P	[B+P S D]		↑			
	B+	[B+ S1 S2 D]	• (S1)+(S2) → (D)	⎍	4		6-30
	B+P	[B+P S1 S2 D]		↑			
	B-	[B− S D]	• (D)-(S) → (D)	⎍	3	●	6-28
	B-P	[B−P S D]		↑			
	B-	[B− S1 S2 D]	• (S1)-(S2) → (D)	⎍	4		6-30
	B-P	[B−P S1 S2 D]		↑			
BCD 8-digit addition and subtraction operations	DB+	[DB+ S D]	• (D+1, D)+(S+1, S) → (D+1, D)	⎍	3		6-32
	DB+P	[DB+P S D]		↑			
	DB+	[DB+ S1 S2 D]	• (S1+1, S1)+(S2+1, S2) → (D+1, D)	⎍	4		6-34
	DB+P	[DB+P S1 S2 D]		↑			
	DB-	[DB− S D]	• (D+1, D)-(S+1, S) → (D+1, D)	⎍	3		6-32
	DB-P	[DB − P S D]		↑			
	DB-	[DB− S1 S2 D]	• (S1+1, S1)-(S2+1, S2) → (D+1, D)	⎍	4		6-34
	DB-P	[DB−P S1 S2 D]		↑			
BCD 4-digit multiplication and division operations	B*	[B* S1 S2 D]	• (S1) * (S2) → (D+1, D)	⎍	4	●	6-36
	B*P	[B*P S1 S2 D]		↑			
	B/	[B/ S1 S2 D]	• (S1)/(S2) → Quotient(D), Remainder (D+1)	⎍	4	●	6-36
	B/P	[B/P S1 S2 D]		↑			
BCD 8-digit multiplication and division operations	DB*	[DB* S1 S2 D]	• (S1+1, S1) * (S2+1, S2) → (D+3, D+2, D+1, D)	⎍	4		6-38
	DB*P	[DB*P S1 S2 D]		↑			
	DB/	[DB/ S1 S2 D]	• (S1+1, S1)/(S2+1, S2) → Quotient (D+1, D), Remainder (D+3, D+2)	⎍	4	●	6-38
	DB/P	[DB/P S1 S2 D]		↑			

Table 2.11 Arithmetic Operation Instructions (Continued)

Category	Instruction Symbols	Symbol	Processing Details	Execution Condition	Number of Basic Steps	Subset	See for Description
Floating decimal point data addition and sub-traction operations	E+	E+ S D	• (D+1, D)+(S+1, S) → (D+1, D)	⎍	3		6-40
	E+P	E+P S D		↑			
	E+	E+ S1 S2 D	• (S1+1, S1)+(S2+1, S2) → (D+1, D)	⎍	4		6-42
	E+P	E+P S1 S2 D		↑			
	E-	E− S D	• (D+1, D)-(S+1, S) → (D+1, D)	⎍	3		6-40
	E-P	E−P S D		↑			
	E-	E− S1 S2 D	• (S1+1, S1)-(S2+1, S2) → (D+1, D)	⎍	4		6-42
	E-P	E−P S1 S2 D		↑			
Floating decimal point data multiplica-tion and division operations	E∗	E∗ S1 S2 D	• (S1+1, S1) ∗ (S2+1, S2) → (D+1, D)	⎍	3		6-44
	E∗P	E∗P S1 S2 D		↑			
	E/	E/ S1 S2 D	• (S1+1, S1)/(S2+1, S2) → Quotient (D+1, D)	⎍	4		6-44
	E/P	E/P S1 S2 D		↑			
BIN block addition and sub-traction operations	BK+	BK+ S1 S2 D n	• Adds data of n points from (S1) and data of n points from (S2) in batch.	⎍	5		6-46
	BK+P	BK+P S1 S2 D n		↑			
	BK-	BK− S1 S2 D n	• Subtracts data of n points from (S1) and data of n points from (S2) in batch.	⎍	5		6-46
	BK-P	BK−P S1 S2 D n		↑			
Character string data combina-tions	$+	$+ S D	• Links character string designated with (S) to character string designated with (D), and stores the result from (D) onward.	⎍	3		6-49
	$+P	$+P S D		↑			
	$+	$+ S1 S2 D	• Links character string designated with (S2) to character string designated with (S1), and stores the result from (D) onward.	⎍	4		6-51
	$+P	$+P S1 S2 D		↑			

Table 2.11 Arithmetic Operation Instructions (Continued)

Category	Instruction Symbols	Symbol	Processing Details	Execution Condition	Number of Basic Steps	Subset	See for Description
BIN data increment	INC	INC D	• (D)+1 → (D)	⊓	2	●	6-53
	INCP	INCP D		↑			
	DINC	DINC D	• (D+1, D)+1 → (D+1, D)	⊓	＊1	●	6-55
	DINCP	DINCP D		↑			
	DEC	DEC D	• (D)-1 → (D)	⊓	2	●	6-53
	DECP	DECP D		↑			
	DDEC	DDEC D	• (D+1, D)-1 → (D+1, D)	⊓	＊1	●	6-55
	DDECP	DDECP D		↑			

REMARKS

1) ＊1:The number of steps may vary depending on the device and type of CPU module being used.

Component	Nomber of basic steps	
High Performance model QCPU Process CPU	(1) When using the following devices only • Word device : Internal device (except for file register ZR) • Bit device : Devices whose device Nos. are multiples of 16, whose digit designation is K8, and which use no index modification. • Constant : No limitations	Note 1) 3
	(2) When using devices other than (1)	Note 2) 2
Basic model QCPU QnCPU	2	Note 2)

Note 1:With High Performance module QCPU, (1) requires more number of steps, while it can process the steps faster, as compared with (2).

Note 2:The number of steps may increase due to the conditions described in Section 3.8.

2.4.3 Data conversion instructions

Table 2.12 Data Conversion Instructions

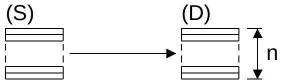
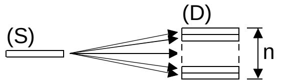
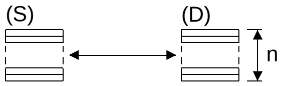
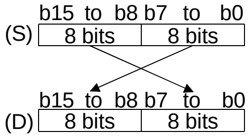
Category	Instruction Symbols	Symbol	Processing Details	Execution Condition	Number of Basic Steps	Subset	See for Description
BCD conversions	BCD	BCD S D	BCD conversion • (S) → (D) ↑ BIN (0 to 9999)	⎍	3	●	6-57
	BCDP	BCDP S D		↑			
	DBCD	DBCD S D	BCD conversion • (S+1, S) → (D+1, D) ↑ BIN (0 to 99999999)	⎍	3	●	6-57
	DBCDP	DBCDP S D		↑			
BIN conversions	BIN	BIN S D	BIN conversion • (S) → (D) ↑ BCD (0 to 9999)	⎍	3	●	6-59
	BINP	BINP S D		↑			
	DBIN	DBIN S D	BIN conversion • (S+1, S) → (D+1, D) ↑ BCD (0 to 99999999)	⎍	3	●	6-59
	DBINP	DBINP S D		↑			
Conversion from BIN to floating decimal point	FLT	FLT S D	Conversion to floating decimal point • (S+1, S) → (D) ↑ BIN (-32768 to 32767)	⎍	3		6-61
	FLTP	FLTP S D		↑			
	DFLT	DFLT S D	Conversion to floating decimal point • (S+1, S) → (D+1, D) ↑ Real number (-2147483648 to 2147483647)	⎍	3		6-61
	DFLTP	DFLTP S D		↑			
Conversion from floating decimal point to BIN	INT	INT S D	Conversion to BIN • (S+1, S) → (D) ↑ Real number (-32768 to 32767)	⎍	3		6-63
	INTP	INTP S D		↑			
	DINT	DINT S D	Conversion to BIN • (S+1, S) → (D+1, D) ↑ Real number (-2147483648 to 2147483647)	⎍	3		6-63
	DINTP	DINTP S D		↑			
Conversion between BIN 16-bit and 32-bit	DBL	DBL S D	Conversion • (S) → (D+1, D) ↑ BIN (-32768 to 32767)	⎍	3		6-65
	DBLP	DBLP S D		↑			
	WORD	WORD S D	Conversion • (S+1, S) → (D) ↑ BIN (-32768 to 32767)	⎍	3		6-66
	WORDP	WORDP S D		↑			
Conversion from BIN to gray code	GRY	GRY S D	Conversion to gray code • (S) → (D) ↑ BIN (-32768 to 32767)	⎍	3		6-67
	GRYP	GRYP S D		↑			
	DGRY	DGRY S D	Conversion to gray code • (S+1, S) → (D+1, D) ↑ BIN (-2147483648 to 2147483647)	↑	3		6-67
	DGRYP	DGRYP S D		↑			

Table 2.12 Data Conversion Instructions (Continued)

Category	Instruction Symbols	Symbol	Processing Details	Execution Condition	Number of Basic Steps	Subset	See for Description
Conversion from gray code to BIN	GBIN	─[GBIN │ S │ D]─	Conversion to BIN data • $\underset{\uparrow \text{Gray code}}{(S)} \longrightarrow (D)$ (-32768 to 32767)	⎍	3		6-69
	GBINP	─[GBINP │ S │ D]─		↑			
	DGBIN	─[DGBIN │ S │ D]─	Conversion to BIN data • $\underset{\uparrow \text{Gray code}}{(S+1, S)} \longrightarrow (D+1, D)$ (-2147483648 to 2147483647)	⎍	3		6-69
	DGBINP	─[DGBINP │ S │ D]─		↑			
Comple-ment to 2	NEG	─[NEG │ D]─	• $\underset{\uparrow \text{BIN data}}{\overline{(D)}} \longrightarrow (D)$	⎍	2		6-71
	NEGP	─[NEGP │ D]─		↑			
	DNEG	─[DNEG │ D]─	• $\underset{\uparrow \text{BIN data}}{\overline{(D+1, D)}} \longrightarrow (D+1, D)$	⎍	2		6-71
	DNEGP	─[DNEGP │ D]─		↑			
	ENEG	─[ENEG │ D]─	• $\underset{\uparrow \text{Real number data}}{\overline{(D+1, D)}} \longrightarrow (D+1, D)$	⎍	2		6-73
	ENEGP	─[ENEGP │ D]─		↑			
Block con-versions	BKBCD	─[BKBCD │ S │ D │ n]─	• Batch converts BIN data n points from (S) to BCD data and stores the result from (D) onward.	⎍	4		6-74
	BKBCDP	─[BKBCDP │ S │ D │ n]─		↑			
	BKBIN	─[BKBIN │ S │ D │ n]─	• Batch converts BCD data n points from (S) to BIN data and stores the result from (D) onward.	⎍	4		6-76
	BKBINP	─[BKBINP │ S │ D │ n]─		↑			

2.4.4 Data transfer instructions

Table 2.13 Data Transfer Instructions

Category	Instruction Symbols	Symbol	Processing Details	Execution Condition	Number of Basic Steps	Subset	See for Description
16-bit data transfer	MOV	MOV S D	• (S) → (D)	⎍	∗4	●	6-78
	MOVP	MOVP S D		↑	∗1		
32-bit data transfer	DMOV	DMOV S D	• (S+1, S) → (D+1, D)	⎍	∗2	●	6-78
	DMOVP	DMOVP S D		↑			
Floating decimal point data transfer	EMOV	EMOV S D	• (S+1, S) → (D+1, D) Real number data	⎍	∗2	● ∗3	6-80
	EMOVP	EMOVP S D		↑			
Character string data transfer	$MOV	$MOV S D	• Transfers character string designated by (S) to device designated by (D) onward.	⎍	3		6-82
	$MOVP	$MOVP S D		↑			
16-bit data negation transfer	CML	CML S D	• ($\overline{S}$) → (D)	⎍	∗1	●	6-84
	CMLP	CMLP S D		↑			
32-bit data negation transfer	DCML	DCML S D	• ($\overline{S+1, S}$) → (D+1, D)	⎍	∗2	●	6-84
	DCMLP	DCMLP S D		↑			
Block transfer	BMOV	BMOV S D n		⎍	4	●	6-87
	BMOVP	BMOVP S D n		↑			
Multiple transfers of same data block	FMOV	FMOV S D n		⎍	4	●	6-89
	FMOVP	FMOVP S D n		↑			
16-bit data exchange	XCH	XCH S D	• (S) ↔ (D)	⎍	3	●	6-91
	XCHP	XCHP S D		↑			
32-bit data exchange	DXCH	DXCH S D	• (S+1, S) ↔ (D+1, D)	⎍	3	●	6-91
	DXCHP	DXCHP S D		↑			
Block data exchange	BXCH	BXCH S D n		⎍	4		6-93
	BXCHP	BXCHP S D n		↑			
Exchange of upper and lower bytes	SWAP	SWAP D	 b15 to b8 b7 to b0 (S) 8 bits 8 bits b15 to b8 b7 to b0 (D) 8 bits 8 bits	⎍	3		6-95
	SWAPP	SWAPP D		↑			

REMARK

1) ＊1:The number of steps may vary depending on the device and type of CPU module being used.

Component	Nomber of basic steps	
High Performance model QCPU Process CPU	(1) When using the following devices only • Word device : Internal device (except for file register ZR) • Bit device : Devices whose device Nos. are multiples of 16, whose digit designation is K8, and which use no index modification. • Constant : No limitations	2
	(2) When using devices other than (1)	Note 2) 3

Note 2:The number of steps may increase due to the conditions described in Section 3.8.

2) ＊2:The number of steps may vary depending on the device and type of CPU module being used.

Component	Nomber of basic steps	
High Performance model QCPU Process CPU	(1) When using the following devices only • Word device : Internal device (except for file register ZR) • Bit device : Devices whose device Nos. are multiples of 16, whose digit designation is K8, and which use no index modification. • Constant : No limitations	Note 1) 2
	(2) When using devices other than (1)	Note 2) 3
Basic model QCPU QnCPU	3	Note 2)

Note 1:With High Performance module QCPU, (1) requires more number of steps, while it can process the steps faster, as compared with (2).
Note 2:The number of steps may increase due to the conditions described in Section 3.8.

3) ＊3 : The subset is effective only with QCPU.

4) ＊4 : The number of steps may vary depending on the device and type of CPU module being used.

Component	Nomber of basic steps	
High Performance model QCPU Process CPU	(1) When using the following devices only • Word device : Internal device (except for file register ZR) • Bit device : Devices whose device Nos. are multiples of 16, whose digit designation is K8, and which use no index modification. • Constant : No limitations	Note 1) 3
	(2) When using devices other than (1)	Note 2) 3
Basic model QCPU	(1) When using the following devices only • Word device : Internal device (except for file register ZR) • Bit device : Devices whose device Nos. are multiples of 16, whose digit designation is K8, and which use no index modification. • Constant : No limitations	Note 1) 2
	(2) When using devices other than (1)	Note 2) 3
QnACPU	3	Note 2)

Note 1:With High Performance module QCPU, (1) requires more number of steps, while it can process the steps faster, as compared with (2).
Note 2:The number of steps may increase due to the conditions described in Section 3.8.

2.4.5 Program branch instruction

Table 2.14 Program Branch Instruction

Category	Instruction Symbols	Symbol	Processing Details	Execution Condition	Number of Basic Steps	Subset	See for Description
Jump	CJ	CJ Pn	• Jumps to Pn when input conditions are met	⎍	2	●	6-96
	SCJ	SCJ Pn	• Jumps to Pn from the scan after the meeting of input condition	⎍	2	●	6-96
	JMP	JMP Pn	• Jumps unconditionally to Pn		2	●	6-96
	GOEND	GOEND	• Jumps to END instruction when input condition is met	⎍	1		6-99

2.4.6 Program execution control instructions

Table 2.15 Program Execution Control Instructions

Category	Instruction Symbols	Symbol	Processing Details	Execution Condition	Number of Basic Steps	Subset	See for Description
Disable interrupts	DI	DI	• Prohibits the running of an interrupt program		1		6-100
Enable interrupts	EI	EI	• Resets interrupt program execution prohibition		1		6-100
Interrupt disable /enable setting	IMASK	IMASK S	• Prohibits or permits interrupts for each interrupt program		2		6-100
Return	IRET	IRET	• Returns to sequence program following an interrupt program		1		6-109

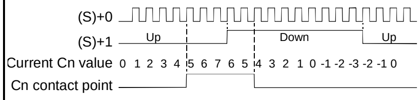
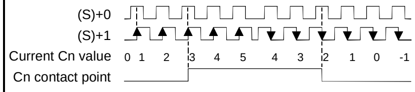
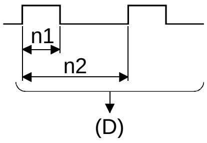
2.4.7 I/O refresh instructions

Table 2.16 I/O Refresh Instructions

Category	Instruction Symbols	Symbol	Processing Details	Execution Condition	Number of Basic Steps	Subset	See for Description
I/O Refresh	RFS	RFS D n	• Refreshes the relevant I/O area during scan	⎍	3		6-111
	RFSP	RFSP D n		↑			

2.4.8 Other convenient instructions

Table 2.17 Other Convenient Instructions

Category	Instruction Symbols	Symbol	Processing Details	Execution Condition	Number of Basic Steps	Subset	See for Description
Up/Down Counter	UDCNT1	UDCNT1 | S | D | n	 (S)+0 (S)+1 Up Down Up Current Cn value 0 1 2 3 4 5 6 7 6 5 4 3 2 1 0 -1 -2 -3 -2 -1 0 Cn contact point	⎍	4		6-113
	UDCNT2	UDCNT2 | S | D | n	 (S)+0 (S)+1 Current Cn value 0 1 2 3 4 5 4 3 2 1 0 -1 Cn contact point	⎍	4		6-115
Teaching timer	TTMR	TTMR | D | n	• (Time that TTMR is ON) ✻n ⟶ (D) n = 0:1, n = 1:10, n = 2:100	⎍	3		6-117
Special timer	STMR	STMR | S | n | D	• The 4 points from the bit device designated by (D) operate as shown below, depending on the ON/OFF status of the input conditions for the STMR instruction: (D)+0: Off delay timer output (D)+1: One shot after off timer output (D)+2: One shot after on timer output (D)+3: On delay timer output		3		6-119
Nearest path control	ROTC	ROTC | S | n1 | n2 | D	• Rotates a rotary table with n1 divisions from the stop position to the position designated by (S+1) by the nearest path.	⎍	5		6-122
Ramp signal	RAMP	RAMP | n1 | n2 | D1 | n3 | D2	• Changes device data designated by D1 from n1 to n2 in n3 scans.	⎍	6		6-124
Pulse density	SPD	SPD | S | n | D	• Counts the pulse input from the device designated by (S) for the duration of time designated by n, and stores the count in the device designated by (D).	⎍	4		6-126
Pulse output	PLSY	PLSY | n1 | n2 | D	• (n1)Hz ⟶ (D) Output n2 times	⎍	4		6-128
Pulse width modulation	PWM	PWM | n1 | n2 | D		⎍	4		6-130
Matrix input	MTR	MTR | S | D1 | D2 | n	• Store 16 times of n lows in the device specified by (S).to the device specified by (D2) in sequence.	⎍	5		6-132

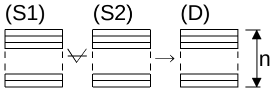
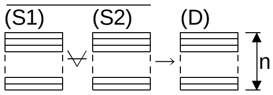
2.5 Application Instructions

2.5.1 Logical operation instructions

Table 2.18 Logical Operation Instructions

Category	Instruction Symbols	Symbol	Processing Details	Execution Condition	Number of Basic Steps	Subset	See for Description
Logical product	WAND	─[WAND S D]─	• (D)∧(S)→(D)	⎍	3	●	7-3
	WANDP	─[WANDP S D]─		↑			
	WAND	─[WAND S1 S2 D]─	• (S1)∧(S2)→(D)	⎍	4	● ∗3	7-5
	WANDP	─[WANDP S1 S2 D]─		↑			
	DAND	─[DAND S D]─	• (D+1, D)∧(S+1, S)→(D+1, D)	⎍	∗1	●	7-3
	DANDP	─[DANDP S D]─		↑			
	DAND	─[DAND S1 S2 D]─	• (S1+1, S1)∧(S2+1, S2)→(D+1, D)	⎍	∗2	● ∗3	7-5
	DANDP	─[DANDP S1 S2 D]─		↑			
	BKAND	─[BKAND S1 S2 D n]─	(S1) ∧ (S2) → (D) n	⎍	5		7-8
	BKANDP	─[BKANDP S1 S2 D n]─		↑			
Logical sum	WOR	─[WOR S D]─	• (D)∨(S)→(D)	⎍	3	●	7-10
	WORP	─[WORP S D]─		↑			
	WOR	─[WOR S1 S2 D]─	• (S1)∨(S2)→(D)	⎍	4	● ∗3	7-12
	WORP	─[WORP S1 S2 D]─		↑			
	DOR	─[DOR S D]─	• (D+1, D)∨(S+1, S)→(D+1, D)	⎍	∗1	●	7-10
	DORP	─[DORP S D]─		↑			
	DOR	─[DOR S1 S2 D]─	• (S1+1, S1)∨(S2+1, S2)→(D+1, D)	⎍	∗2	● ∗3	7-12
	DORP	─[DORP S1 S2 D]─		↑			
	BKOR	─[BKOR S1 S2 D n]─	(S1) ∨ (S2) → (D) n	⎍	5		7-14
	BKORP	─[BKORP S1 S2 D n]─		↑			
Exclusive OR	WXOR	─[WXOR S D]─	• (D)⊻(S)→(D)	⎍	3	●	7-16
	WXORP	─[WXORP S D]─		↑			
	WXOR	─[WXOR S1 S2 D]─	• (S1)⊻(S2)→(D)	⎍	4	● ∗3	7-18
	WXORP	─[WXORP S1 S2 D]─		↑			

Table 2.18 Logical Operation Instructions (Continued)

Category	Instruction Symbols	Symbol	Processing Details	Execution Condition	Number of Basic Steps	Subset	See for Description
Exclusive OR	DXOR	DXOR S D	• $(D+1, D)\veebar(S+1, S)\rightarrow(D+1, D)$	⎍	∗1	●	7-16
	DXORP	DXORP S D		↑			
	DXOR	DXOR S1 S2 D	• $(S1+1, S1)\veebar(S2+1, S2)\rightarrow(D+1, D)$	⎍	∗2	● ∗3	7-18
	DXORP	DXORP S1 S2 D		↑			
	BKXOR	BKXORP S1 S2 D n		⎍	5		7-20
	BKXORP	BKXORP S1 S2 D n		↑			
NON exclusive logical sum	WXNR	WXNR S D	• $\overline{(D)\veebar(S)}\rightarrow(D)$	⎍	3	●	7-22
	WXNRP	WXNRP S D		↑			
	WXNR	WXNR S1 S2 D	• $\overline{(S1)\veebar(S2)}\rightarrow(D)$	⎍	4	● ∗3	7-26
	WXNRP	WXNRP S1 S2 D		↑			
	DXNR	DXNR S D	• $\overline{(D+1, D)\veebar(S+1, S)}\rightarrow(D+1, D)$	⎍	∗1	●	7-22
	DXNRP	DXNRP S D		↑			
	DXNR	DXNR S1 S2 D	• $\overline{(S1+1, S1)\veebar(S2+1, S2)}\rightarrow(D+1, D)$	⎍	∗2	● ∗3	7-26
	DXNRP	DXNRP S1 S2 D		↑			
	BKXNR	BKXNR S1 S2 D n		⎍	5		7-28
	BKXNRP	BKXNRP S1 S2 D n		↑			

REMARK

1) ∗1:The number of steps may vary depending on the device and type of CPU module being used.

Component	Nomber of basic steps	
High Performance model QCPU Process CPU	(1) When using the following devices only • Word device : Internal device (except for file register ZR) • Bit device : Devices whose device Nos. are multiples of 16, whose digit designation is K8, and which use no index modification. • Constant : No limitations	Note 1) 5
	(2) When using devices other than (1)	Note 2) 3
Basic model QCPU QnCPU	3	Note 2)

Note 1:With High Performance module QCPU, (1) requires more number of steps, while it can process the steps faster, as compared with (2).

Note 2:The number of steps may increase due to the conditions described in Section 3.8.

2) ＊2:The number of steps may vary depending on the device and type of CPU module being used.

Component	Nomber of basic steps	
High Performance model QCPU Process CPU	(1) When using the following devices only • Word device : Internal device (except for file register ZR) • Bit device : Devices whose device Nos. are multiples of 16, whose digit designation is K8, and which use no index modification. • Constant : No limitations	Note 1) 6
	(2) When using devices other than (1)	Note 2) 4
Basic model QCPU QnCPU	4	Note 2)

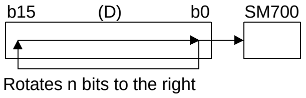
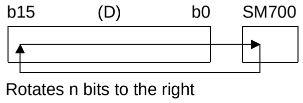
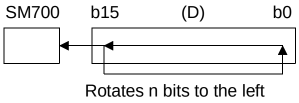
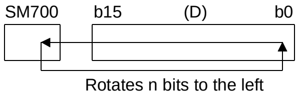
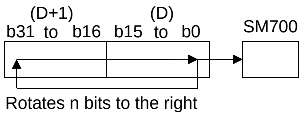
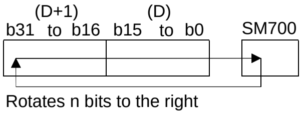
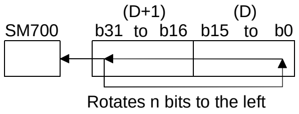
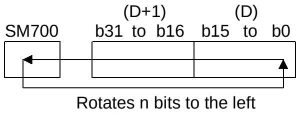
Note 1:With High Performance module QCPU, (1) requires more number of steps, while it can process the steps faster, as compared with (2).

Note 2:The number of steps may increase due to the conditions described in Section 3.8.

3) ＊3 : The subset is effective only with QCPU.

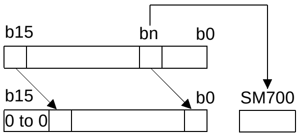
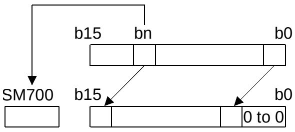
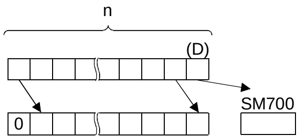
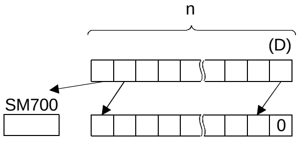
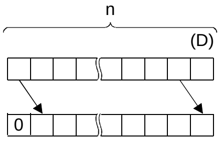
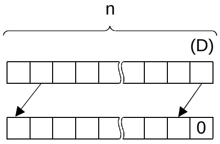
2.5.2 Rotation instructions

Table 2.19 Rotation Instructions

Category	Instruction Symbols	Symbol	Processing Details	Execution Condition	Number of Basic Steps	Subset	See for Description
Right rotation	ROR	ROR D n			3	●	7-30
	RORP	RORP D n					
	RCR	RCR D n			3	●	7-30
	RCRP	RCRP D n					
Left rotation	ROL	ROL D n			3	●	7-32
	ROLP	ROLP D n					
	RCL	RCL D n			3	●	7-32
	RCLP	RCLP D n					
Right rotation	DROR	DROR D n	 (D+1) (D) b31 to b16 b15 to b0 SM700 Rotates n bits to the right		3	●	7-34
	DRORP	DRORP D n					
	DRCR	DRCR D n	 (D+1) (D) b31 to b16 b15 to b0 SM700 Rotates n bits to the right		3	●	7-34
	DRCRP	DRCRP D n					
Left rotation	DROL	DROL D n	 (D+1) (D) SM700 b31 to b16 b15 to b0 Rotates n bits to the left		3	●	7-36
	DROLP	DROLP D n					
	DRCL	DRCL D n	 (D+1) (D) SM700 b31 to b16 b15 to b0 Rotates n bits to the left		3	●	7-36
	DRCLP	DRCLP D n					

2.5.3 Shift instructions

Table 2.20 Shift Instructions

Category	Instruction Symbols	Symbol	Processing Details	Execution Condition	Number of Basic Steps	Subset	See for Description
n-bit shift	SFR	SFR D n		⎍	3	●	7-38
	SFRP	SFRP D n		↑			
	SFL	SFL D n		⎍	3	●	7-38
	SFLP	SFLP D n		↑			
1-bit shift	BSFR	BSFR D n		⎍	3		7-40
	BSFRP	BSFRP D n		↑			
	BSFL	BSFL D n		⎍	3		7-40
	BSFLP	BSFLP D n		↑			
1-word shift	DSFR	DSFR D n		⎍	3	●	7-42
	DSFRP	DSFRP D n		↑			
	DSFL	DSFL D n		⎍	3	●	7-42
	DSFLP	DSFLP D n		↑			

2.5.4 Bit processing instructions

Table 2.21 Bit processing instructions

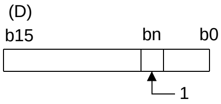
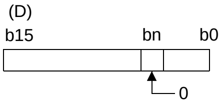
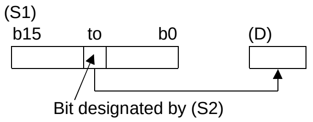
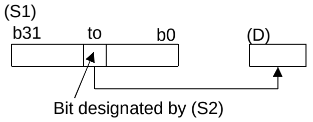
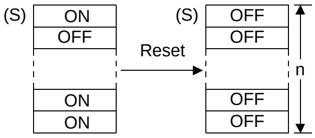
Category	Instruction Symbols	Symbol	Processing Details	Execution Condition	Number of Basic Steps	Subset	See for Description
Bit set / reset	BSET	BSET D n		⎍	3	●	7-44
	BSETP	BSETP D n		↑			
	BRST	BRST D n		⎍	3	●	7-44
	BRSTP	BRSTP D n		↑			

Table 2.21 Bit processing Instructions (Continued)

Category	Instruction Symbols	Symbol	Processing Details	Execution Condition	Number of Basic Steps	Subset	See for Description
Bit tests	TEST	TEST S1 S2 D			4		7-46
	TESTP	TESTP S1 S2 D					
	DTEST	DTEST S1 S2 D			4		7-46
	DTESTP	DTESTP S1 S2 D					
Batch reset of bit devices	BKRST	BKRST S n	 (S) ON OFF ON ON (S) OFF OFF OFF OFF Reset n		3		7-48
	BKRSTP	BKRSTP S n					

2.5.5 Data processing instructions

Table 2.22 Data Processing Instructions

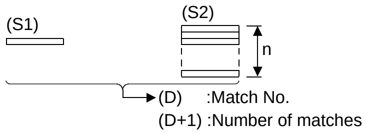
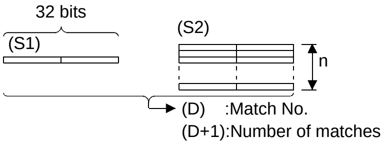
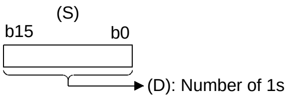
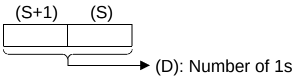
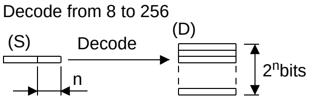
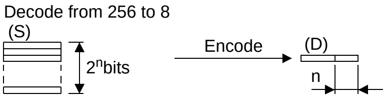
Category	Instruction Symbols	Symbol	Processing Details	Execution Condition	Number of Basic Steps	Subset	See for Description
Data searches	SER	SER S1 S2 D n	 (S1) (S2) n (D) :Match No. (D+1) :Number of matches		5		7-50
	SERP	SERP S1 S2 D n					
	DSER	DSER S1 S2 D n	 32 bits (S1) (S2) n (D) :Match No. (D+1):Number of matches		5		7-50
	DSERP	DSERP S1 S2 D n					
Bit checks	SUM	SUM S D			3	●	7-54
	SUMP	SUMP S D					
	DSUM	DSUM S D			3	●	7-54
	DSUMP	DSUMP S D					
Decode	DECO	DECO S D n			4		7-56
	DECOP	DECOP S D n					
Encode	ENCO	ENCO S D n			4		7-58
	ENCOP	ENCOP S D n					

Table 2.22 Data Processing Instructions (Continued)

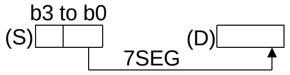
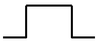
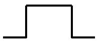
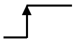
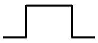
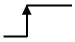
Category	Instruction Symbols	Symbol	Processing Details	Execution Condition	Number of Basic Steps	Subset	See for Description
7-segment decode	SEG	SEG S D	 b3 to b0 (S) 7SEG (D)	⎍	3	●	7-60
	SEGP	SEGP S D		↑			
Separating and linking	DIS	DIS S D n	• Separates 16-bit data designated by (S) into 4-bit units, and stores at the lower 4 bits of n points from (D). (n ≤ 4)	⎍	4		7-62
	DISP	DISP S D n		↑			
	UNI	UNI S D n	• Links the lower 4 bits of n points from the device designated by (S) and stores at the device designated by (D). (n ≤ 4)	⎍	4		7-64
	UNIP	UNIP S D n		↑			
	NDIS	NDIS S1 D S2	• Separates the data at the devices below that designated by (S1) into bits designated below (S2) and stores in sequence from the device designated by (D).	⎍	4		7-66
	NDISP	NDISP S1 D S2		↑			
	NUNI	NUNI S1 D S2	• Links the data at the devices below that designated by (S1) in the bits designated below (S2) and stores in sequence from the device designated by (D).	⎍			
	NUNIP	NUNIP S1 D S2		↑			
	WTOB	WTOB S D n	• Breaks n-points of 16-bit data from the device designated by (S) into 8-bit units, and stores in sequence at the device designated by (D).	⎍	4		7-71
	WTOBP	WTOBP S D n		↑			
	BTOW	BTOW S D n	• Links the lower 8 bits of 16-bit data of n-points from the device designated by (S) into 16-bit units, and stores in sequence at the device designated by (D).	⎍			
	BTOWP	BTOWP S D n		↑			
Search	MAX	MAX S D n	• Searches the data of n-points from the device designated by (S) in 16-bit units, and stores the maximum value at the device designated by (D).	⎍	4		7-75
	MAXP	MAXP S D n		↑			
	MIN	MIN S D n	• Searches the data of n-points from the device designated by (S) in 16-bit units, and stores the minimum value at the device designated by (D).	⎍			7-77
	MINP	MINP S D n		↑			
	DMAX	DMAX S D n	• Searches the data of 2＊n-points from the device designated by (S) in 32-bit units, and stores the maximum value at the device designated by (D).	⎍	4		7-75
	DMAXP	DMAXP S D n		↑			
	DMIN	DMIN S D n	• Searches the data of 2＊n-points from the device designated by (S) in 32-bit units, and stores the minimum value at the device designated by (D).	⎍			7-77
	DMINP	DMINP S D n		↑			

Table 2.22 Data Processing Instructions (Continued)

Category	Instruction Symbols	Symbol	Processing Details	Execution Condition	Number of Basic Steps	Subset	See for Description
Sort	SORT	SORT S1 n S2 D1 D2 • S2:Number of comparisons made during one run • D1:Device to turn ON when sort is completed • D2:For system use	• Sorts data of n-points from device designated by (S1) in 16-bit units. (n x (n-1)/2 scans required)		6		7-80
	DSORT	DSORT S1 n S2 D1 D2 • S2:Number of comparisons made during one run • D1:Device to turn ON when sort is completed • D2:For system use	• Sorts data of 2＊n-points from device designated by (S1) in 32-bit units. (n x (n+1)/2 scans required)				
Total value calcula-tions	WSUM	WSUM S D n	• Adds 16 bit BIN data of n points from the device specified by (S), and stores it in the device specified by (D).		4		7-83
	WSUMP	WSUMP S D n					
	DWSUM	DWSUM S D n	• Adds 32 bit BIN data of n points from the device specified by (S), and stores it in the device specified by (D).				7-85
	DWSUMP	DWSUMP S D n					

2.5.6 Structure creation instructions

Table 2.23 Structure Creation Instructions

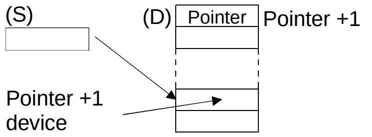
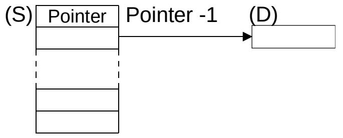
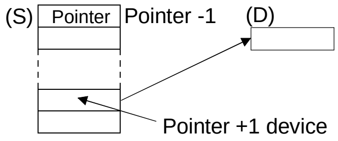
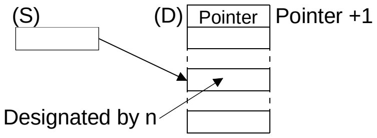
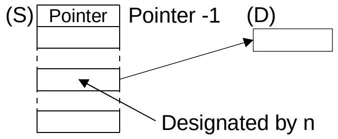
Category	Instruction Symbols	Symbol	Processing Details	Execution Condition	Number of Basic Steps	Subset	See for Description
Number of repeats	FOR	FOR n	• Executes n times between FOR and NEXT		2		7-87
	NEXT	NEXT			1		
	BREAK	BREAK D Pn	• Forcibly ends the execution of the FOR to NEXT cycle and jumps pointer to Pn.	⎍	3		7-89
	BREAKP	BREAKP D Pn		↑			
Sub-routine program calls	CALL	CALL Pn S1 to Sn	• Executes sub-routine program Pn when input condition is met. (S1 to Sn are arguments sent to sub-routine program. 0 ≤ n ≤ 5)	⎍	∗1 2 + n		7-91
	CALLP	CALLP Pn S1 to Sn		↑			
	RET	RET	• Returns from sub-routine program		1		7-94
	FCALL	FCALL Pn S1 to Sn	• Performs non-execution processing on sub-routine program Pn if input conditions have not been met	⊔	∗1 2 + n		7-95
	FCALLP	FCALLP Pn S1 to Sn		↓			
	ECALL	ECALL ∗ Pn S1 to Sn ∗: Program Name	• Executes sub-routine program Pn from within designated program name when input condition is met. (S1 to Sn are arguments sent to sub-routine program. 0 ≤ n ≤ 5)	⎍	∗2 3 + n		7-99
	ECALLP	ECALLP ∗ Pn S1 to Sn ∗: Program Name		↑			
	EFCALL	EFCALL ∗ Pn S1 to Sn ∗: Program Name	• Performs non-execution processing of sub-routine program Pn from within designated program name if input condition is not met.	⎍	∗2 3 + n		7-102
	EFCALLP	EFCALLP ∗ Pn S1 to Sn ∗: Program Name		↑			
	COM	COM	• Performs link refresh and general data processing.		1		7-016
Fixed index modification	IX	IX S Device modification ladder	• Conducts index modification for individual devices used in device modification ladder.		2		7-112
	IXEND	IXEND			1		
	IXDEV	IXDEV	• Stores modification value used for index modification performed between IX and IXEND in the device below that designated by (D).		1		7-120
	IXSET	IXSET Pn D Designates modification value			3		

∗1 : n indicates number of arguments for sub-routine program.

∗2 : n indicates the total of the number of arguments used in the sub-routine program and the number of program name steps. The number of program name steps is calculated as "number of characters in the program / 2" (decimal fraction is rounded up).

2.5.7 Table operation instructions

Table 2.24 Table Operation Instructions

Category	Instruction Symbols	Symbol	Processing Details	Execution Condition	Number of Basic Steps	Subset	See for Description
Table processing	FIFW	FIFW S D		⎍	3		7-125
	FIFWP	FIFWP S D		↑			
	FIFR	FIFR S D		⎍	3		7-127
	FIFRP	FIFRP S D		↑			
	FPOP	FPOP S D		⎍	3		7-129
	FPOPP	FPOPP S D		↑			
	FINS	FINS S D n		⎍	4		7-131
	FINSP	FINSP S D n		↑			
	FDEL	FDEL S D n		⎍	4		7-131
	FDELP	FDELP S D n		↑			

2.5.8 Buffer memory access instructions

Table 2.25 Buffer Memory Access Instructions

Category	Instruction Symbols	Symbol	Processing Details	Execution Condition	Number of Basic Steps	Subset	See for Description
Data read	FROM	FROM n1 n2 D n3	• Reads data in 16-bit units from special function module	⎍	5		7-134
	FROMP	FROMP n1 n2 D n3		↑			
	DFRO	DFRO n1 n2 D n3	• Reads data in 32-bit units from special function module	⎍	5		7-134
	DFROP	DFROP n1 n2 D n3		↑			
Data write	TO	TO n1 n2 S n3	• Writes data in 16-bit units to special function module	⎍	5		7-137
	TOP	TOP n1 n2 S n3		↑			
	DTO	DTO n1 n2 S n3	• Writes data in 32-bit units to special function module	⎍	5		7-137
	DTOP	DTOP n1 n2 S n3		↑			

2.5.9 Display instructions

Table 2.26 Display Instructions

Category	Instruction Symbols	Symbol	Processing Details	Execution Condition	Number of Basic Steps	Subset	See for Description
ASCII print	PR	✻ SM701 When OFF PR S D	• Outputs ASCII code of 8 points (16 characters) from device designated by (S) to output module.	↑	3		7-140
	PR	✻ SM701 When ON PR S D	• Outputs ASCII code from device designated by (S) to 00H to output module.				
	PRC	PRC S D	• Converts comments from device designated by (S) to ASCII code and outputs to output module.				7-143
Display	LED	LED S	• Displays ASCII code of 8 points (16 characters) from the device designated by (S) at the LED display device on the front of the CPU.	↑	2		7-148
	LEDC	LEDC S	• Displays the comments from the device designated by (S) at the LED display device on the front of the CPU module.				7-150
Reset	LEDR	LEDR	• Resets annunciator and display unit display.	↑	1		7-152

2.5.10 Debugging and failure diagnosis instructions

Table 2.27 Debugging and Failure Diagnosis Instructions

Category	Instruction Symbols	Symbol	Processing Details	Execution Condition	Number of Basic Steps	Subset	See for Description
Checks	CHKST	CHKST	• CHK instruction is executed when CHKST is executable. • Jumps to the step following the CHK instruction when CHKST is in a non-executable status		1		7-155
	CHK	CHK Check Condition	• During normal conditions → SM80: OFF, SD80: 0 • During abnormal conditions → SM80: ON, SD80: Failure No.				
	CHKCIR	CHKCIR	• Starts update in ladder pattern being checked by CHK instruction		1		7-159
	CHKEND	CHKEND	• Ends update in ladder pattern being checked by CHK instruction				
Status latch	SLT	SLT	• Executes status latch	↑	1		7-167
	SLTR	SLTR	• Resets status latch to enable re-execution				
Sampling trace	STRA	STRA	• Applies trigger to sampling trace	↑	1		7-169
	STRAR	STRAR	• Resets sampling trace to enable re-execution				
Program trace	PTRA	PTRA	• Applies trigger to program trace	↑	1		7-171
	PTRAR	PTRAR	• Resets program trace to enable re-execution				
	PTRAEXE	PTRAEXE	• Executes program trace	⊓	1		7-171
	PTRAEXE P	PTRAEXEP		↑			

2.5.11 Character string processing instructions

Table 2.28 Character String Processing Instructions

Category	Instruction Symbols	Symbol	Processing Details	Execution Condition	Number of Basic Steps	Subset	See for Description
BIN to decimal ASCII	BINDA	BINDA S D	• Converts 1-word BIN value designated by (S) to a 5-digit, decimal ASCII value, and stores it at the word device designated by (D).	⎍	3		7-173
	BINDAP	BINDAP S D		↑			
	DBINDA	DBINDA S D	• Converts 2-word BIN value designated by (S) to a 10-digit, decimal ASCII value, and stores it at word devices following the word device number designated by (D).	⎍	3		7-173
	DBINDAP	DBINDAP S D		↑			
BIN to hexa-decimal ASCII	BINHA	BINHA S D	• Converts 1-word BIN value designated by (S) to a 4-digit, hexadecimal ASCII value, and stores it at a word device following the word device number designated by (D).	⎍	3		7-176
	BINHAP	BINHAP S D		↑			
	DBINHA	DBINHA S D	• Converts 2-word BIN value designated by (S) to an 8-digit, hexadecimal ASCII value, and stores it at word devices following the word device number designated by (D).	⎍	3		7-176
	DBINHAP	DBINHAP S D		↑			
BCD to decimal ASCII	BCDDA	BCDDA S D	• Converts 1-word BCD value designated by (S) to a 4-digit, decimal ASCII value, and stores it at a word device following the word device number designated by (D).	⎍	3		7-179
	BCDDAP	BCDDAP S D		↑			
	DBCDDA	DBCDDA S D	• Converts 2-word BCD value designated by (S) to an 8-digit, decimal ASCII value, and stores it at word devices following the word device number designated by (D).	⎍	3		7-179
	DBCDDAP	DBCDDAP S D		↑			
Decimal ASCII to BIN	DABIN	DABIN S D	• Converts a 5-digit, decimal ASCII value designated by (S) to a 1-word BIN value, and stores it at a word device number designated by (D).	⎍	3		7-182
	DABINP	DABINP S D		↑			
	DDABIN	DDABIN S D	• Converts a 10-digit, decimal ASCII value designated by (S) to a 2-word BIN value, and stores it at a word device number designated by (D).	⎍	3		7-182
	DDABINP	DDABINP S D		↑			
Hexadeci-mal ASCII to BIN	HABIN	HABIN S D	• Converts a 4-digit, hexadecimal ASCII value designated by (S) to a 1-word BIN value, and stores it at a word device number designated by (D).	⎍	3		7-185
	HABINP	HABINP S D		↑			
	DHABIN	DHABIN S D	• Converts an 8-digit, hexadecimal ASCII designated by (S) value to a 2-word BIN value, and stores it at a word device number designated by (D).	⎍	3		7-185
	DHABINP	DHABINP S D		↑			

Table 2.28 Character String Processing Instructions (Continued)

Category	Instruction Symbols	Symbol	Processing Details	Execution Condition	Number of Basic Steps	Subset	See for Description
Decimal ASCII to BCD	DABCD	DABCD S D	• Converts a 4-digit, decimal ASCII value designated by (S) to a 1-word BCD value, and stores it at a word device number designated by (D).	⎍	3		7-187
	DABCDP	DABCDP S D		↑			
	DDABCD	DDABCD S D	• Converts an 8-digit, decimal ASCII designated by (S) value to a 2-word BCD value, and stores it at a word device number designated by (D).	⎍	3		7-187
	DDABCDP	DDABCDP S D		↑			
Device comment read operation	COMRD	COMRD S D	• Stores comment from device designated by (S) at a device designated by (D).	⎍	3		7-190
	COMRDP	COMRDP S D		↑			
Character string length detection	LEN	LEN S D	• Stores data length (number of characters) in character string designated by (S) at a device designated by (D).	⎍	3		7-194
	LENP	LENP S D		↑			
BIN to decimal character string	STR	STR S1 S2 D	• Converts a 1-word BIN value designated by (S2) to a decimal character string with the total number of digits and the number of decimal fraction digits designated by (S1) and stores them at a device designated by (D).	⎍	4		7-196
	STRP	STRP S1 S2 D		↑			
	DSTR	DSTR S1 S2 D	• Converts a 2-word BIN value designated by (S2) to a decimal character string with the total number of digits and the number of decimal fraction digits designated by (S1) and stores them at a device designated by (D).	⎍	4		7-196
	DSTRP	DSTRP S1 S2 D		↑			
Decimal character string to BIN	VAL	VAL S D1 D2	• Converts a character string including decimal point designated by (S) to a 1-word BIN value and the number of decimal fraction digits, and stores them at devices designated by (D1) and (D2).	⎍	4		7-202
	VALP	VALP S D1 D2		↑			
	DVAL	DVAL S D1 D2	• Converts a character string including decimal point designated by (S) to a 2-word BIN value and the number of decimal fraction digits, and stores them at devices designated by (D1) and (D2).	⎍	4		7-202
	DVALP	DVALP S D1 D2		↑			
Floating decimal point to character string	ESTR	ESTR S1 S2 D	• Converts floating decimal point data designated by (S1) to character string, and stores them in a device designated by (D).	⎍	4		7-207
	ESTRP	ESTRP S1 S2 D		↑			
Character string to floating decimal point	EVAL	EVAL S D	• Converts character string designated by (S) to floating decimal point data, and stores them in a device designated by (D).	⎍	3		7-214
	EVALP	EVALP S D		↑			

Table 2.28 Character String Processing Instructions (Continued)

Category	Instruction Symbols	Symbol	Processing Details	Execution Condition	Number of Basic Steps	Subset	See for Description
Hexadecimal BIN to ASCII	ASC	ASC S D n	• Converts 1-word BIN values of the device number and later designated by (S) to ASCII, and stores only n characters of them at the device number designated by (D).	⎍	4		7-218
	ASCP	ASCP S D n		↑			
ASCII to hexadecimal BIN	HEX	HEX S D n	• Converts only n ASCII characters of the device number and later designated by (S) to BIN values, and stores them at the device number designated by (D).	⎍	4		7-220
	HEXP	HEXP S D n		↑			
Character string processing	RIGHT	RIGHT S D n	• Stores n characters from the end of a character string designated by (S) at the device designated by (D).	⎍	4		7-222
	RIGHTP	RIGHTP S D n		↑			
	LEFT	LEFT S D n	• Stores n characters from the beginning of a character string designated by (S) at the device designated by (D).	⎍			
	LEFTP	LEFTP S D n		↑			
	MIDR	MIDR S1 D S2	• Stores the designated number of characters in the character string designated by (S1) from the position designated by (S2) at the device designated by (D).	⎍	4		7-225
	MIDRP	MIDRP S1 D S2		↑			
	MIDW	MIDW S1 D S2	• Stores the designated number of characters in the character string designated by (S1) from the position designated by (S2) at the device designated by (D).	⎍			
	MIDWP	MIDWP S1 D S2		↑			
	INSTR	INSTR S1 S2 D n	• Searches character string (S1) from the nth character of character string (S2), and stores matched positions at (D).	⎍	5		7-229
	INSTRP	INSTRP S1 S2 D n		↑			
Floating decimal point to BCD	EMOD	EMOD S1 S2 D	• Converts floating decimal point data (S1) to BCD data with number of decimal fraction digits designated by (S2) , and stores at device designated by (D).	⎍	4		7-231
	EMODP	EMODP S1 S2 D		↑			
BCD to floating decimal point data	EREXP	EREXP S1 S2 D	• Converts BCD data (S1) to floating decimal point data with the number of decimal fraction digits designated by (S2), and stores at device designated by (D).	⎍	4		7-233
	EREXPP	EREXPP S1 S2 D		↑			

2.5.12 Special function instructions

Table 2.29 Special Function Instructions

Category	Instruction Symbols	Symbol	Processing Details	Execution Condition	Number of Basic Steps	Subset	See for Description
Trigono-metric functions (Floating decimal point data)	SIN	SIN S D	• Sin (S+1, S) → (D+1, D)	⎍	3		7-235
	SINP	SINP S D		↑			
	COS	COS S D	• Cos (S+1, S) → (D+1, D)	⎍	3		7-237
	COSP	COSP S D		↑			
	TAN	TAN S D	• Tan (S+1, S) → (D+1, D)	⎍	3		7-239
	TANP	TANP S D		↑			
	ASIN	ASIN S D	• Sin^{-1} (S+1, S) → (D+1, D)	⎍	3		7-241
	ASINP	ASINP S D		↑			
	ACOS	ACOS S D	• Cos^{-1} (S+1, S) → (D+1, D)	⎍	3		7-243
	ACOSP	ACOSP S D		↑			
	ATAN	ATAN S D	• Tan^{-1} (S+1, S) → (D+1, D)	⎍	3		7-245
	ATANP	ATANP S D		↑			
Conversion between angles and radians	RAD	RAD S D	• (S+1, S) → (D+1, D) Conversion from angles to radians	⎍	3		7-247
	RADP	RADP S D		↑			
	DEG	DEG S D	• (S+1, S) → (D+1, D) Conversion from radians to angles	⎍	3		7-249
	DEGP	DEGP S D		↑			
Square root	SQR	SQR S D	• $\sqrt{(S+1, S)}$ → (D+1, D)	⎍	3		7-251
	SQRP	SQRP S D		↑			
Exponent operations	EXP	EXP S D	• e$^{(S+1,\ S)}$ → (D+1, D)	⎍	3		7-253
	EXPP	EXPP S D		↑			
Natural logarithms	LOG	LOG S D	• Log e (S+1, S) → (D+1, D)	⎍	3		7-255
	LOGP	LOGP S D		↑			
Random number generation	RND	RND D	• Generates a random number (from 0 to less than 32767) and stores it at the device designated by (D).	⎍	2		7-257
	RNDP	RNDP D		↑			
Random number series update	SRND	SRND S	• Updates random number series according to the 16-bit BIN data stored in the device designated by (S).	⎍			
	SRNDP	SRNDP S		↑			

Table 2.29 Special Function Instructions (Continued)

Category	Instruction Symbols	Symbol	Processing Details	Execution Condition	Number of Basic Steps	Subset	See for Description
Square root	BSQR	─[BSQR | S | D]─	• $\sqrt{(S)}$ ⟶ (D)+0 Integer part +1 Decimal fraction part	⎍	3		7-259
	BSQRP	─[BSQRP | S | D]─		↑			
	BDSQR	─[BDSQR | S | D]─	• $\sqrt{(S+1, S)}$ ⟶ (D)+0 Integer part +1 Decimal fraction part	⎍	3		7-259
	BDSQRP	─[BDSQRP | S | D]─		↑			
Trigono-metric function	BSIN	─[BSIN | S | D]─	• Sin (S) ⟶ (D)+0 Sign +1 Integer part +2 Decimal fraction part	⎍	3		7-262
	BSINP	─[BSINP | S | D]─		↑			
	BCOS	─[BCOS | S | D]─	• Cos (S) ⟶ (D)+0 Sign +1 Integer part +2 Decimal fraction part	⎍	3		7-264
	BCOSP	─[BCOSP | S | D]─		↑			
	BTAN	─[BTAN | S | D]─	• Tan (S) ⟶ (D)+0 Sign +1 Integer part +2 Decimal fraction part	⎍	3		7-266
	BTANP	─[BTANP | S | D]─		↑			
	BASIN	─[BASIN | S | D]─	• Sin^{-1} (S) ⟶ (D)+0 Sign +1 Integer part +2 Decimal fraction part	⎍	3		7-268
	BASINP	─[BASINP | S | D]─		↑			
	BACOS	─[BACOS | S | D]─	• Cos^{-1} (S) ⟶ (D)+0 Sign +1 Integer part +2 Decimal fraction part	⎍	3		7-270
	BACOSP	─[BACOSP | S | D]─		↑			
	BATAN	─[BATAN | S | D]─	• Tan^{-1} (S) ⟶ (D)+0 Sign +1 Integer part +2 Decimal fraction part	⎍	3		7-272
	BATANP	─[BATANP | S | D]─		↑			

2.5.13 Data control instructions

Table 2.30 Data Control Instructions

Category	Instruction Symbols	Symbol	Processing Details	Execution Condition	Number of Basic Steps	Subset	See for Description
Upper and lower limit controls	LIMIT	LIMIT S1 S2 S3 D	• When (S3) < (S1) .... Store value of (S1) at (D) • When (S1) ≤ (S3) ≤ (S2) .... Store value of (S3) at (D) • When (S2) < (S3) .... Store value of (S2) at (D)	⎍	5		7-274
	LIMITP	LIMITP S1 S2 S3 D		↑			
	DLIMIT	DLIMIT S1 S2 S3 D	• When ((S3)+1, (S3)) < ((S1)+1, S1) ...Store value of ((S1)+1, (S1)) at ((D)+1, (D)) • When ((S1)+1, (S1)) ≤ ((S3)+1, (S3))< (S2+1, S2) ...Store value of ((S3)+1, (S3)) at ((D)+1, (D)) • When ((S2), (S2)+1) < ((S3), (S3)+1) ...Store value of ((S2)+1, (S2)) at ((D)+1, (D))	⎍	5		7-274
	DLIMITP	DLIMITP S1 S2 S3 D		↑			
Dead band controls	BAND	BAND S1 S2 S3 D	• When (S1) ≤ (S3) ≤ (S2) ...0 → (D) • When (S3) < (S1) ....(S3)-(S1) → (D) • When (S2) < (S3) ....(S3)-(S2) → (D)	⎍	5		7-277
	BANDP	BANDP S1 S2 S3 D		↑			
	DBAND	DBAND S1 S2 S3 D	• When ((S1)+1, (S1)) ≤ ((S3)+1, (S3)) ≤ ((S2)+1, (S2)) ...0 → ((D)+1, (D)) • When ((S3)+1, (S3)) < ((S1)+1, (S1)) ..((S3)+1, (S3)) - ((S1)+1, (S1)) → ((D)+1, (D)) • When ((S2)+1, (S2)) < ((S3)+1, (S3)) ..((S3)+1, (S3)) - ((S2)+1, (S2)) → ((D)+1, (D))	⎍	5		7-277
	DBANDP	DBANDP S1 S2 S3 D		↑			
Zone controls	ZONE	ZONE S1 S2 S3 D	• When (S3) = 0 ....0 → (D) • When (S3) > 0 ....(S3)+(S2) → (D) • When (S3) < 0 ....(S3)-(S1) → (D)	⎍	5		7-280
	ZONEP	ZONEP S1 S2 S3 D		↑			
	DZONE	DZONE S1 S2 S3 D	• When ((S3)+1, (S3)) = 0 ...0 → ((D)+1, (D)) • When ((S3)+1, (S3)) > 0 ...((S3)+1, (S3))+((S2)+1, (S2)) → ((D)+1, (D)) • When ((S3)+1, (S3)) < 0 ...((S3)+1, (S3)) + ((S1)+1, (S1)) → ((D)+1, (D))	⎍	5		7-280
	DZONEP	DZONEP S1 S2 S3 D		↑			

2.5.14 Switching instructions

Table 2.31 Switching Instructions

Category	Instruction Symbols	Symbol	Processing Details	Execution Condition	Number of Basic Steps	Subset	See for Description
Block number designa-tions	RSET	RSET │ S	• Converts extension file register block number to number designated by (S).	⊓	2		7-283
	RSETP	RSETP │ S		↑			
File set	QDRSET	QDRSET │ File Name	• Sets file names used as file registers.	⊓	* 2 + n		7-285
	QDRSETP	QDRSETP │ File Name		↑			
	QCDSET	QCDSET │ File Name	• Sets file names used as comment files.	⊓	* 2 + n		7-287
	QCDSETP	QCDSETP │ File Name		↑			

* : n ([number of file name characters] / 2) indicates a step. (Decimal fractions are rounded up.)

2.5.15 Clock instructions

Table 2.32 Clock Instructions

Category	Instruction Symbols	Symbol	Processing Details	Execution Condition	Number of Basic Steps	Subset	See for Description
Read/write clock data	DATERD	─┤ DATERD D ├─	• (Clock device) → (D)+0 Year +1 Month +2 Day +3 Hour +4 Minute +5 Sec. +6 Day of week	⎍	2		7-289
	DATERDP	─┤ DATERDP D ├─		↑			
	DATEWR	─┤ DATEWR S ├─	• (D)+0 Year → (Clock device) +1 Month +2 Day +3 Hour +4 Minute +5 Sec. +6 Day of week	⎍	2		7-293
	DATEWRP	─┤ DATEWRP S ├─		↑			
Clock data addition/ subtraction	DATE+	─┤ DATE+ S1 S2 D ├─	(S1) Hour, Minute, Sec. + (S2) Hour, Minute, Sec. → (D) Hour, Minute, Sec.	⎍	4		7-297
	DATE+P	─┤ DATE+P S1 S2 D ├─		↑			
	DATE-	─┤ DATE− S1 S2 D ├─	(S1) Hour, Minute, Sec. - (S2) Hour, Minute, Sec. → (D) Hour, Minute, Sec.	⎍	4		7-299
	DATE-P	─┤ DATE−P S1 S2 D ├─		↑			
Clock data translation	SECOND	─┤ SECOND S D ├─	(S) Hour, Minute, Sec. → (D) Sec. (lower level), Sec. (upper level)	⎍	3		7-301
	SECONDP	─┤ SECONDP S D ├─		↑			
	HOUR	─┤ HOUR S D ├─	(S) Sec. (lower level), Sec. (upper level) → (D) Hour, Minute, Sec.	⎍			
	HOURP	─┤ HOURP S D ├─		↑			

2.5.16 Peripheral device instructions

Table 2.33 Peripheral Device Instructions

Category	Instruction Symbols	Symbol	Processing Details	Execution Condition	Number of Basic Steps	Subset	See for Description
Input/ output to peripheral devices	MSG	MSG S	• Stores message designated by (S) at QnACPU. This message is displayed at the peripheral device	⎍	2		7-303
	PKEY	PKEY D	• Data input from the peripheral device is stored at device designated by (D).	⎍	2		7-305

2.5.17 Program instructions

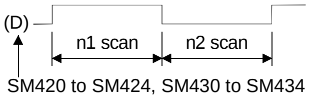
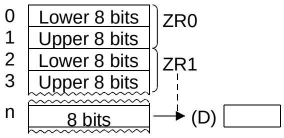
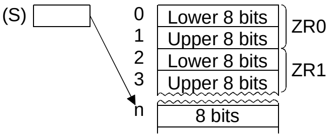
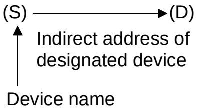
Table 2.34 Program Instructions

Category	Instruction Symbols	Symbol	Processing Details	Execution Condition	Number of Basic Steps	Subset	See for Description
Switching program execution statuses	PSTOP	PSTOP Program Name	• Places designated program in standby status	⎍	*2+n		7-308
	PSTOPP	PSTOPP Program Name		↑			
	POFF	POFF Program Name	• Turns OUT instruction coil of designated program OFF, and places program in standby status.	⎍	*2+n		7-309
	POFFP	POFFP Program Name		↑			
	PSCAN	PSCAN Program Name	• Registers designated program as scan execution program.	⎍	*2+n		7-311
	PSCANP	PSCANP Program Name		↑			
	PLOW	PLOW Program Name	• Registers designated program as low-speed execution program.	⎍	*2+n		7-313
	PLOWP	PLOWP Program Name		↑			

* : n ([number of program name characters] / 2) indicates a step. (Decimal fractions are rounded up.)

2.5.18 Other instructions

Table 2.35 Other Instructions

Category	Instruction Symbols	Symbol	Processing Details	Execution Condition	Number of Basic Steps	Subset	See for Description
WDT reset	WDT	WDT	• Resets watchdog timer during sequence program	⎍	1		7-315
	WDTP	WDTP		↑			
Timing clock	DUTY	DUTY n1 n2 D		↑	4		7-317
Direct read/write operations in 1-byte units	ZRRDB	ZRRDB n D	 0 Lower 8 bits 1 Upper 8 bits 2 Lower 8 bits 3 Upper 8 bits ZR0 ZR1 n 8 bits (D)	⎍	3		7-319
	ZRRDBP	ZRRDBP n D		↑			
	ZRWRB	ZRWRB n S	 (S) 0 Lower 8 bits 1 Upper 8 bits 2 Lower 8 bits 3 Upper 8 bits ZR0 ZR1 n 8 bits	⎍	3		7-321
	ZRWRBP	ZRWRBP n S		↑			
	ADRSET	ADRSET S D	 (S) (D) Indirect address of designated device Device name	⎍	3		7-323
	ADRSETP	ADRSETP S D		↑			
Numerical key input from keyboard	KEY	KEY S n D1 D2	• Takes in ASCII data for 8 points of input unit designated by (S), converts to hexadecimal value following device number designated by D1, and stores.	⎍	5		7-324
Batch save of index register	ZPUSH	ZPUSH D	• Saves the contents of index registers Z0 to Z15 to a location starting from the device designated by D.	⎍	2		7-328
	ZPUSHP	ZPUSHP D		↑			
Batch recovery of index register	ZPOP	ZPOP D	• Reads the data stored in the location starting from the device designated by D to index registers Z0 toZ15.	⎍			
	ZPOPP	ZPOPP D		↑			
Batch write operation to E^2PROM file register	EROMWR	EROMWR S D1 n D2	• Writes a batch of data to E^2PROM file register.	⎍	5		7-332
	EROMWRP	EROMWRP S D1 n D2		↑			

2.5.19 Instructions for data link

Table 2.36 Instructions for Data Link

Category	Instruction Symbols	Symbol	Processing Details	Execution Condition	Number of Basic Steps	Subset	See for Description
Network refresh	ZCOM	J.ZCOM Jn	Refreshes the designated network.	⎍	5		8-6
		JP.ZCOM Jn		↑			
		G.ZCOM Un		⎍			
		GP.ZCOM Un		↑			
QnA link instruction: Reading data from another station	READ	J.READ Jn S1 S2 D1 D2 G.READ Jn S1 S2 D1 D2	Reads the word device data of another station to host station.	⎍	9		8-12
		JP.READ Jn S1 S2 D1 D2 GP.READ Un S1 S2 D1 D2		↑			
	SREAD	J.SREAD Jn S1 S2 D1 D2 D3 G.SREAD Un S1 S2 D1 D2 D3		⎍	10		8-18
		JP.SREAD Jn S1 S2 D1 D2 D3 GP.SREAD Un S1 S2 D1 D2 D3		↑			
QnA link instruction: Writing data to other stations	WRITE	J.WRITE Jn S1 S2 D1 D2 G.WRITE Un S1 S2 D1 D2	Writes the data of host station to the word device of other stations.	⎍	10		8-24
		JP.WRITE Jn S1 S2 D1 D2 GP.WRITE Un S1 S2 D1 D2		↑			
	SWRITE	J.SWRITE Jn S1 S2 D1 D2 D3 G.SWRITE Un S1 S2 D1 D2 D3		⎍	11		8-31
		JP.SWRITE Jn S1 S2 D1 D2 D3 GP.SWRITE Un S1 S2 D1 D2 D3		↑			
QnA link instruction: Sending data	SEND	J.SEND Jn S1 S2 D1 G.SEND Un S1 S2 D1	Sends data (message) to other stations.	⎍	8		8-38
		JP.SEND Jn S1 S2 D1 GP.SEND Un S1 S2 D1		↑			
QnA link instruction: Receiving data	RECV	J.RECV Jn S1 S2 D1 G.RECV Un S1 S2 D1	Receives data (message) sent to the host station.	⎍	8		8-46
		JP.RECV Jn S1 S2 D1 GP.RECV Un S1 S2 D1		↑			
QnA link instruction: Transient requests from other stations	REQ	J.REQ Jn S1 S2 D1 D2 G.REQ Un S1 S2 D1 D2	Sends a transient request to other stations and executes it.	⎍	8		8-52
		JP.REQ Jn S1 S2 D1 D2 GP.REQ Un S1 S2 D1 D2		↑			

Table 2.36 Instructions for Data Link (Continued)

Category	Instruction Symbols	Symbol	Processing Details	Execution Condition	Number of Basic Steps	Subset	See for Description
QnA link instruction: Reading data from special function modules at remote I/O stations	ZNFR	JP.ZNFR Jn S1 S2 D1	Reads data from the special function modules at remote I/O stations.	↑	8		8-64
		GP.ZNFR Un S1 S2 D1		↑			
QnA link instruction: Writing data to special function module at remote I/O station	ZNTO	J.ZNTO Jn S1 S2 D	Writes data to the special function module at remote I/O station	⊓	8		8-69
		JP.ZNTO Jn S1 S2 D		↑			
		G.ZNTO Un S1 S2 D		⊓			
		GP.ZNTO Un S1 S2 D		↑			
A-series compatible link instruc-tion: Reading device data from other stations	ZNRD	J.ZNRD Jn n1 D1 S n2 D2	Reads the word device data of other station to host station.	⊓	32		8-74 8-78
		JP.ZNRD Jn n1 D1 S n2 D2		↑			
A-series compatible link instruc-tion: Writing device data to other stations	ZNWR	J.ZNWR Jn n1 D1 S n2 D2	Writes the data of host station to the word device of other stations.	⊓	32		8-81 8-85
		JP.ZNWR Jn n1 D1 S n2 D2		↑			
A-series compatible link instruc-tion: Reading data from special function module at remote I/O station.	RFRP	G.RFRP Un n1 D1 n2 D2	Reads data from the special function module at remote I/O station.	⊓	11		8-88
		GP.RFRP Un n1 D1 n2 D2		↑			
A-series compatible link instruc-tion: Writing data to special function modules at remote I/O stations.	RTOP	G.RTOP Un n1 D1 n2 D	Writes data to the special function module at remote I/O station.	⊓	11		8-92
		GP.RTOP Un n1 D1 n2 D		↑			
Reading routing information	RTREAD	Z.RTREAD n D	Reads data set at routing parameters.	⊓	7		8-96
		ZP.RTREAD n D		↑			
Registering routing information	RTWRITE	Z.RTWRITE n S	Writes routing data to the area designated by routing parameters.	⊓	8		8-100
		ZP.RTWRITE n S		↑			

2.5.20 QCPU instructions

Table 2.37 QCPU Instructions

Category	Instruction Symbols	Symbol	Processing Details	Execution Condition	Number of Basic Steps	Subset	See for Description
Reading module information	UNIRD	UNIRD n1 D n2	• Reads the module information stored in the area starting from the I/O No. designated by (n) by the points designated by (n2), and stores it in the area starting from the device designated by (d).	⎍	4		9-2
	UNIRDP	UNIRDP n1 D n2		↑			
Trace set	TRACE	TRACE	• Stores trace data set at a peripheral device to trace file in IC memory card by the designated number when SM800, SM801, and SM802 turns ON.	↑	1		9-5
Trace reset	TRACER	TRACER	• Resets the data set by TRACE instruction.	↑	1		9-5
Writing data to designated file	SP.FWRITE	SP.FWRITE U0 S0 D0 S1 S2 D1	• Writes data to the designated file.	↑	11		9-7
Reading data from designated file	SP.FREAD	SP.FREAD U0 S0 D0 S1 S2 D1	• Reads data from the designated file.	↑	11		9-15
Loading program from memory	PLOADP	PLOADP S D	• Transfers the program stored in a memory card or standard memory (other than drive 0) to drive 0 and places the program in standby status.	↑	3		9-26
Unloading program from program memory	PUNLOADP	PUNLOADP S D	• Deletes the standby program stored in standard memory (drive 0).	↑	3		9-28
Load + unload	PSWAPP	PSWAPP S1 S2 D	• Deletes standby program stored in standard memory (drive 0) designated by (S1). Then, transfers the program stored in a memory card or standard memory (other than drive 0) designated by (S2) to drive 0 and places it in standby status.	↑	4		9-30
High-speed block transfer of file register	RBMOV	RBMOV S D n	• Transfers n points of 16-bit data from the device designated by (S) to the location starting from the device designated by (D).	⎍	4		9-32
	RBMOVP	RBMOVP S D n		↑			
Write to host station CPU shared memory	S. TO	S.TO n1 n2 n3 n4 D	• Writes the device data of the host station to the shared memory area of the host station CPU module.	⎍	5		9-35
	SP. TO	SP.TO n1 n2 n3 n4 D		↑			
Read from another station CPU shared memory	FROM	FROM n1 n2 D n3	• Reads device data from the CPU shared memory area of another station CPU module to the host station.	⎍	5		9-37
	FROMP	FROMP n1 n2 D n3		↑			
Automatic refresh of CPU shared memory	COM	COM	• Performs the automatic refresh of the intelligent function module, general data processing, and the automatic refresh of the CPU shared memory.		1		9-39

2.5.21 Redundant system instructions (For Q4ARCPU)

Table 2.38 Redundant system instructions (For Q4ARCPU)

Category	Instruction Symbols	Symbol	Processing Details	Execution Condition	Number of Basic Steps	Subset	See for Description
Operation mode setting during CPU start up	S.STMODE	S.STMODE | S1 | S2	• Designates the operation mode at (S1) whether to clear the Q4ARCPU devices before startup or not to clear them before startup when the power supply is turned on for CPU startup.		9		10-2
Operation mode setting instructions during CPU switch	S.CGMODE	S.CGMODE | S	• Designates the operation mode at (S1) whether to clear the Q4ARCPU devices before startup or not to clear them before startup when control is switched from the control system to the standby system.		7		10-4
Data tracking	S.TRUCK	S.TRUCK | S	• Conducts device memory tracking in accordance with the parameter block data contents stored in the area starting from the device designated by (S) during END processing.		6		10-6
Buffer memory batch refresh	S.SPREF	S.SPREF | S	• Batch reads/writes the contents of special function module buffer memory in accordance with the contents of parameter block data stored in the area starting from the device designated by (S).		6		10-10

3. CONFIGURATION OF INSTRUCTIONS

3.1 Configuration of Instructions

Most CPU module instructions consist of an instruction part and a device part.

- Instruction part.....Indicates the function of the instruction
- Device part ....Indicates the data that is to be used with the instruction.

The device part is classified into source data, destination data, and number of devices.

(1) Source Ⓢ

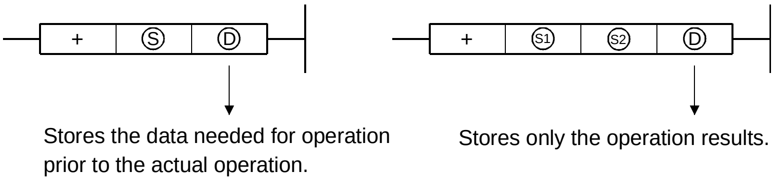
(a) Source is the data used for operations.

(b) The following source types are available, depending on the designated device:

- Constants.....................................Designates the numeric value to be used in the operation.
  This is set when the program is written, and cannot be changed during the execution of the program.
  Constants should be indexed when using them as variable data.
- Bit devices and Word devices.....Designates the device that stores the data to be used for the operation.
  Data must be stored in the designated device until when the operation is executed.
  By changing the data stored in a designated device during program execution, the data to be used in the instruction can be changed.

(2) Destination Ⓓ

(a) The destination stores the data after the operation has been conducted.
However, some instructions require storing the data to be used in an operation at the destination prior to the operation execution.
Example: An addition instruction involving BIN 16-bit data

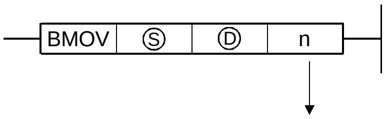

+
Ⓢ
Ⓓ
Stores the data needed for operation prior to the actual operation.
+
S1
S2
Ⓓ
Stores only the operation results.

(b) A device for the data storage must always be set to the destination.

(3) Number of devices and number of transfers (n)

(a) The number of devices and number of transfers designate the numbers of devices and transfers used by instructions involving multiple devices.

Example: Block transfer instruction

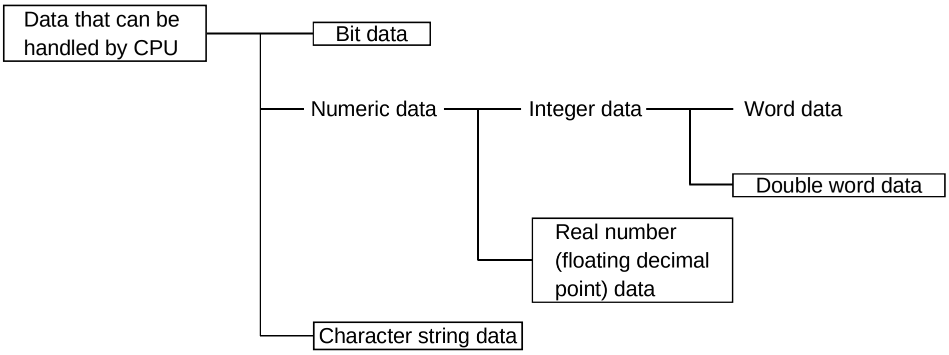


Designates the number of transfers used by a BMOV instruction

(b) The number of devices or number of transfers can be set between 0 and 32767. However, if the number is 0, the instruction will be a no-operation instruction.

3.2 Designating Data

The following five types of data can be used with CPU module instructions:

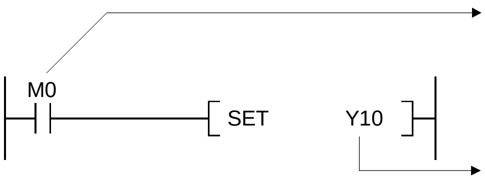


3.2.1 Using bit data

Bit data is data used in one-bit units, such as for contact points or coils. "Bit devices" and "Bit designated word devices" can be used as bit data.

(1) When using bit devices

Bit devices are designated in one-point units.

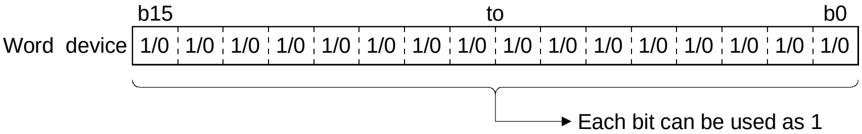


The 1-point M0 is a bit device

The 1-point Y10 is a bit device

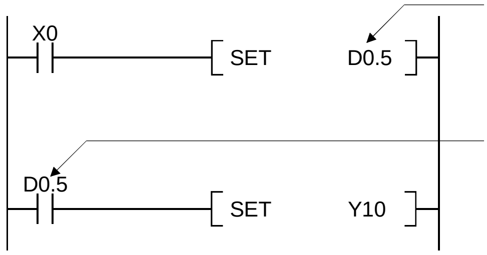
(2) Using word devices

(a) Word devices enable the use of a designated bit number 1/0 as bit data by the designation of that bit number.



Each bit can be used as 1 for ON and 0 for OFF.

(b) Word device bit designation is done by designating "Word Device . Bit No." .
(Designation of bit numbers is done in hexadecimal.)
For example, bit 5 (b5) of D0 is designated as D0.5, and bit 10 (b10) of D0 is designated as D0.A.
However, there can be no bit designation for timers (T), retentive timers (ST), counters (C) or index register (Z). (Example Z0.0 is not available)



Word device bit designation. (Bit 5 (b5) of D0 designated as ON (1).)

Word device bit designation. (Turned ON or OFF according to the 1/O status of bit 5 (b5) of D0.)

3.2.2 Using word (16 bits) data

Word data is 16-bit numeric data used by basic instructions and application instructions.
The following two types of word data can be used with CPU module:

- Decimal constants..........................K-32768 to K32767
- Hexadecimal constants.................H0000 to HFFFF

Word devices and bit devices designated by digit can be used as word data.
For direct access input (DX) and direct access output (DY), word data cannot be designated by digit output (DY). (For details of direct access input and direct access output, refer to the User's Manual (Function Explanation, Program Fundamentals) of the CPU module in use, or the QnACPU Programming Manual (Fundamentals).).

(1) When using bit devices

(a) Bit devices can deal with word data when digits are designated.
Digit designation of bit devices is done by designating "Number of digits Initial number of bit device" .Digit designation of bit devices can be done in 4-point (4-bit) units, and designation can be made for K1 to K4.
(For link direct devices, designation is done by "J Network No. \ Digit designation Initial number of bit device" . When X100 to X10F are designated for Network No.2, it is done by J2\K4X100.)
For example, if X0 is designated for digit designation, the following points would be designated:

- K1X0.........The 4 points X0 to X3 are designated
- K2X0.........The 8 points X0 to X7 are designated
- K3X0.........The 12 points X0 to XB are designated
- K4X0.........The 16 points X0 to XF are designated

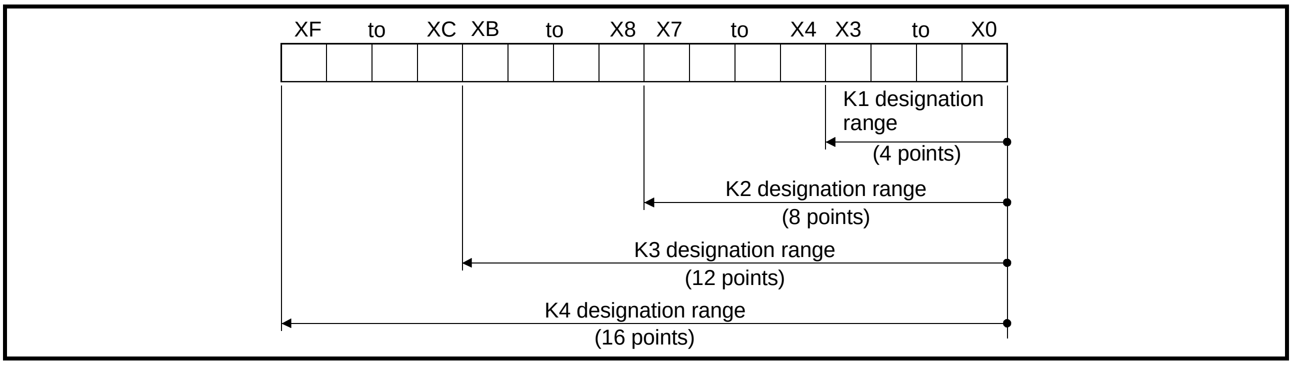


Fig 3.1 Digit Designation Setting Range for 16-Bit Instruction

(b) In cases where digit designation has been made at the source Ⓢ, the numeric values shown in Table 3.1 are those which can be dealt with as source data.

Table 3.1 List of Numeric Values that Can Be Dealt with as Digit Designation

Number of Digits Designated	With 16-Bit Instruction
K1 (4 points)	0 to 15
K2 (8 points)	0 to 255
K3 (12 points)	0 to 4095
K4 (16 points)	-32768 to 32767

In cases where the source is a bit device designated by digit designation, and the destination is a word device, the word device for the destination becomes 0 following the bit designated by digit designation at the source.

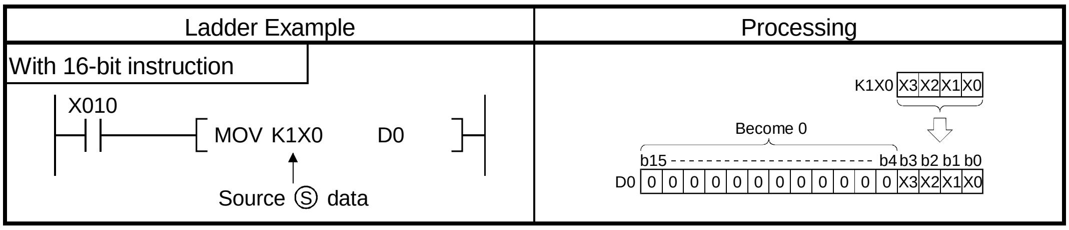


Fig 3.2 Ladder Example and Processing Conducted

(c) In cases where digit designation is made at the destination Ⓓ, the number of points designated are used as the destination. Bit devices below the number of points designated as digits do not change.

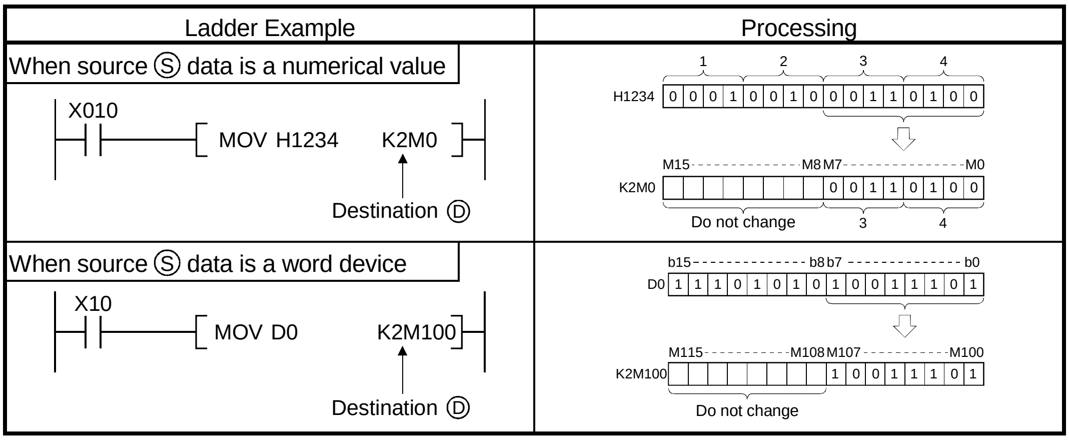
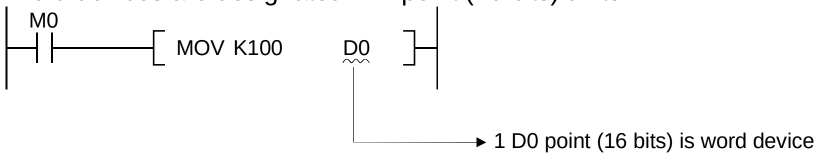


Fig 3.3 Ladder Example and Processing Conducted

(2) When using word devices

Word devices are designated in 1-point (16 bits) units.



POINTS

(1) When digit designation processing is conducted, a random value can be used for the bit device initial device number.
(2) Digit designation cannot be made for the direct device designation DX and DY.

3.2.3 Using double word data (32 bits)

Double word data is 32-bit numerical data used by basic instructions and application instructions.

The two types of double word data that can be dealt with by CPU module are as follows:

- Decimal constants...............K-2147483648 to K2147483647
- Hexadecimal constants.......H00000000 to HFFFFFFFF

Word devices and bit devices designated by digit designation can be used as double word data.

For direct access input (DX) and direct access output (DY), designation of double word data is not possible by digit designation.

(1) When using bit devices

(a) Digit designation can be used to enable a bit device to deal with double word data.

Digit designation of bit devices is done by designating "Number of digits Initial number of bit device".

Digit designation of bit devices can be done in 4-point (4-bit) units, and designation can be made for K1 to K8.

(For link direct devices, designation is done by "J Network No. \ Digit designation Initial number of bit device" . When X100 to X11F are designated for Network No.2, it is done by J2\K8X100.)

For example, if X0 is designated for digit designation, the following points would be designated:

- K1X0.........The 4 points X0 to X3 are designated
- K2X0.........The 8 points X0 to X7 are designated
- K3X0.........The 12 points X0 to XB are designated
- K4X0.........The 16 points X0 to XF are designated
- K5X0.........The 20 points X0 to X13 are designated
- K6X0.........The 24 points X0 to X17 are designated
- K7X0.........The 28 points X0 to X1B are designated
- K8X0.........The 32 points X0 to X1F are designated

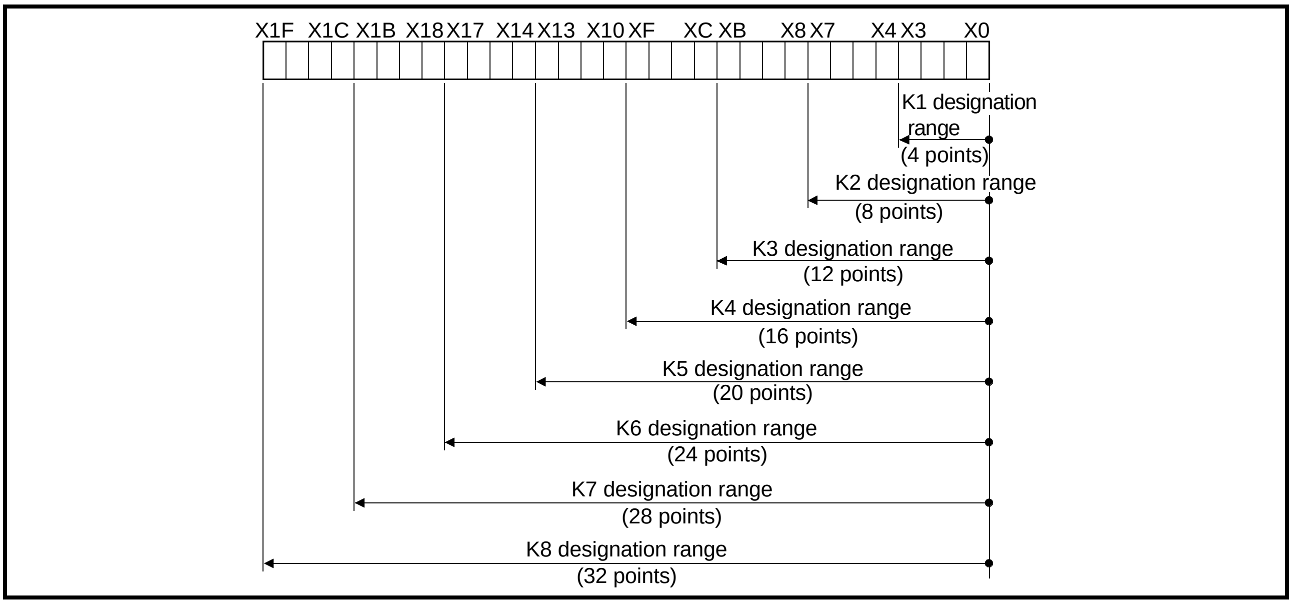

X1F
X1C
X1B
X18
X17
X14
X13
X10
XF
XC
XB
X8
X7
X4
X3
X0
K1 designation range (4 points)
K2 designation range (8 points)
K3 designation range (12 points)
K4 designation range (16 points)
K5 designation range (20 points)
K6 designation range (24 points)
K7 designation range (28 points)
K8 designation range (32 points)

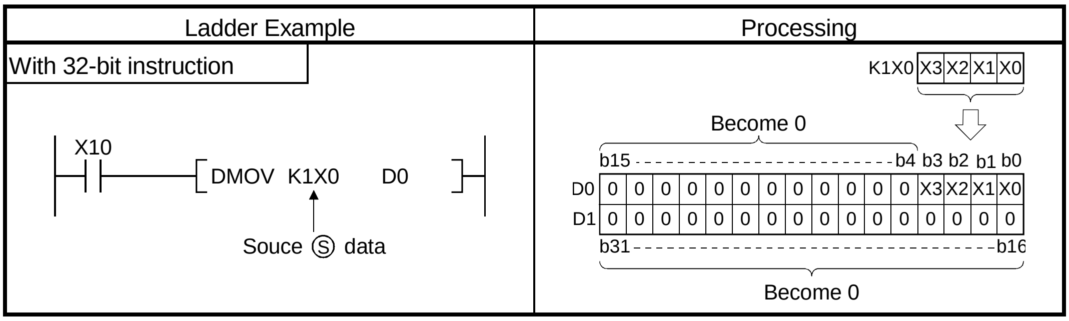
Fig 3.4 Digit Designation Setting Range for 32-Bit Instructions

(b) In cases where digit designation has been made at the source Ⓢ, the numeric values shown in Table 3.2 are those which can be dealt with as source data.

Table 3.2 List of Numeric Values that Can Be Dealt with as Digit Designation

Number of Digits Designated	With 32 bit Instructions	Number of Digits Designated	With 32 bit Instructions
K1 (4 points)	0 to 15	K5 (20 points)	0 to 1048575
K2 (8 points)	0 to 255	K6 (24 points)	0 to 16777215
K3 (12 points)	0 to 4095	K7 (28 points)	0 to 268435455
K4 (16 points)	0 to 65535	K8 (32 points)	-2147483648 to 2147483647

In cases where the source is a bit device designated by digit designation, and the destination is a word device, the word device for the destination becomes 0 following the bit designated by digit designation at the source.



D0	0	0	0	0	0	0	0	0	0	0	0	0	X3	X2	X1	X0
D1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fig 3.5 Ladder Example and Processing Conducted

(c) In cases where digit designation is made at the destination Ⓓ, the number of points designated are used as the destination.
Bit devices after the number of points designated as digits do not change.

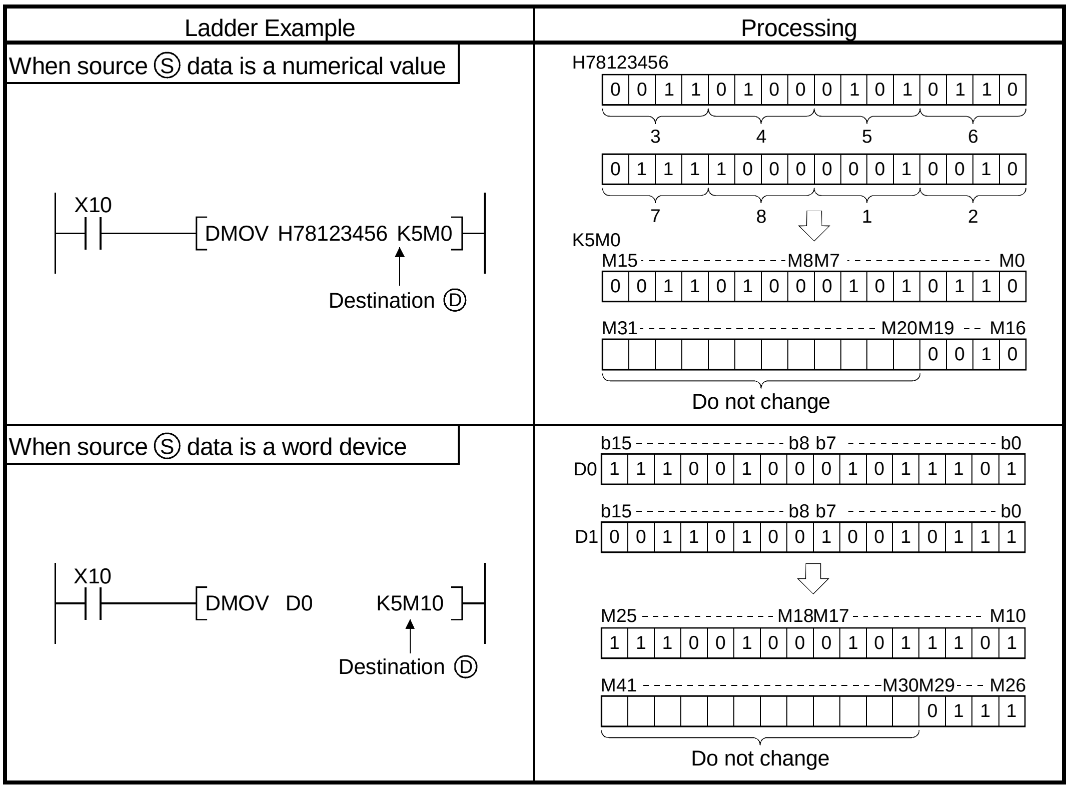
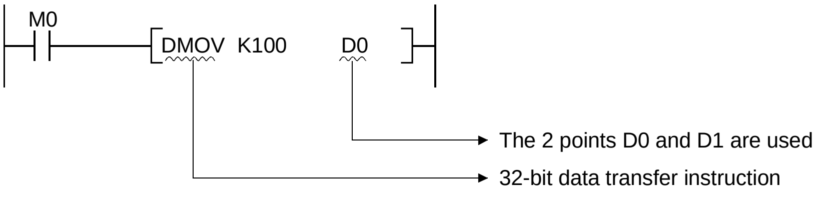


Fig 3.6 Ladder Example and Processing Conducted

POINTS

(1) When digit designation processing is conducted, a random value can be used for the bit device initial device number.
(2) Digit designation cannot be made for the direct device designation DX and DY.

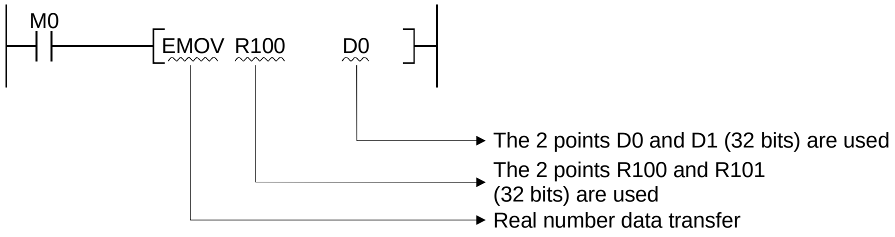
(2) When using word devices
A word device designates devices used by the lower 16 bits of data.
A 32-bit instruction uses (designation device number) and (designation device number + 1).


M0
DMOV K100 D0
The 2 points D0 and D1 are used
32-bit data transfer instruction

3.2.4 Using real number data

Real number data is 32-bit floating decimal point data used with basic instructions and application instructions.
Only word devices are capable of storing real number data.

Instructions which deal with real numbers designate devices which are used for the lower 16 bits of data. Real numbers are stored in the 32 bits which make up (designated device number) and (designated device number + 1).

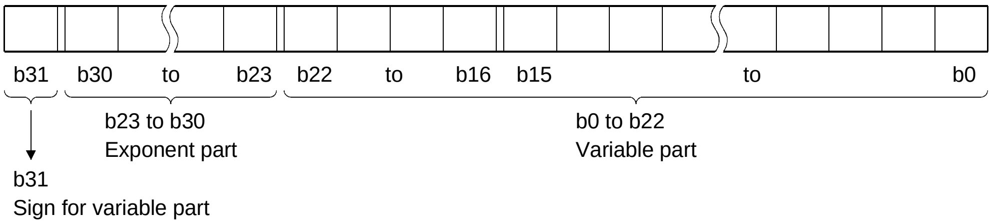

M0
EMOV
R100
D0
The 2 points D0 and D1 (32 bits) are used
The 2 points R100 and R101 (32 bits) are used
Real number data transfer

REMARK

1) In sequence programs, real numbers are designated by E ⬚ .

Floating decimal point data uses two word devices and is expressed in the following manner:
1. [Variable part] × 2 [exponent part]

The bit configuration and meaning of the internal representation of floating decimal point data is as follows:



- Sign for variable part The sign for the variable part is represented at b31.
  0: Positive
  1: Negative
- Exponent part The n of 2^n is represented from b23 to b30.
  Depending on the BIN value of b23 to b30, the value of n is as follows.

b23 to b30	FF_H	FE_H	FD_H			81_H	80_H	$7F_H$	$7E_H$			02_H	01_H	00_H
n	Nonnumeric	127	126			2	1	0	-1			-125	-126	Nonnumeric

- Exponent part The 23 bits from b0 to b22, represents the XXXXXX... at binary 1.XXXXXX....

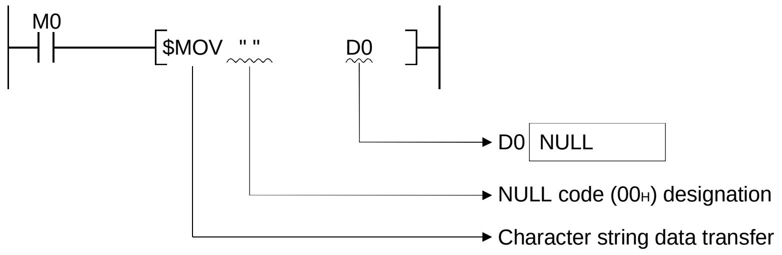
POINT

- The CPU module floating decimal point data can be monitored using the monitoring function of a peripheral device.
- When this is expressed as 0, all data from b0 to b31 will be 0.
- The setting range of real numbers is 0 and $\pm 2^{-126} \leq$ | value | $< 2^{128}$.

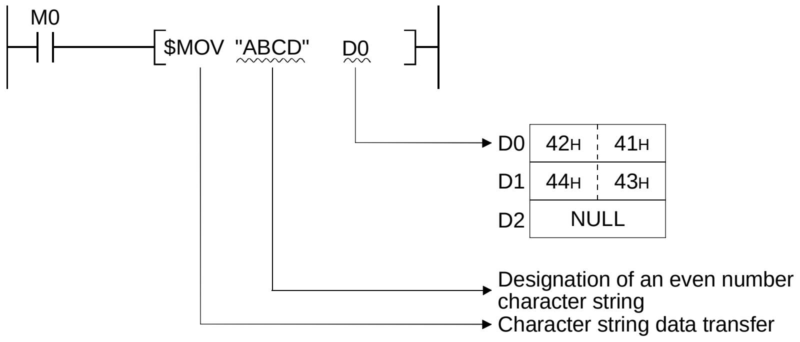
3.2.5 Using character string data

Character string data is character data used by basic instructions and application instructions.
It encompasses all data from the designated character to the NULL code (00_H).

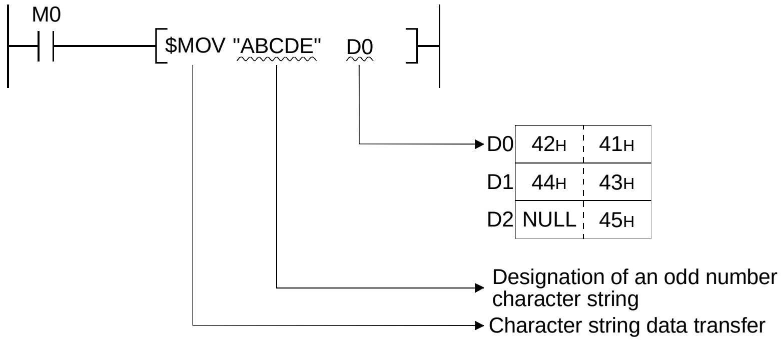
(1) When designated character is the NULL code.
One word is used to store the NULL code.


M0
$MOV " " D0
D0 NULL
NULL code (00_H) designation
Character string data transfer

(2) When character string is even
Uses (number of characters/2 + 1) words, and stores character string and NULL code.
For example, if "ABCD" is transferred to D0, the character string ABCD is stored at D0 and D1, and the NULL code is stored at D2.


M0
$MOV "ABCD" D0
D0 42_H 41_H
D1 44_H 43_H
D2 NULL
Designation of an even number character string
Character string data transfer

(3) When number of characters is odd
Uses (number of characters/2) words (rounds up decimal fractions) and stores the character string and NULL code.
For example, if "ABCDE" is transferred to D0, the character string (ABCDE) and the NULL code are stored from D0 to D2.

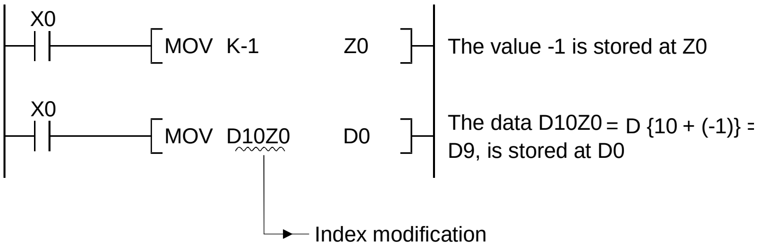

M0
$MOV "ABCDE" D0
D0 42_H 41_H
D1 44_H 43_H
D2 NULL 45_H
Designation of an odd number character string
Character string data transfer

3.3 Index Modification

(1) Index modification

(a) Index modification is an indirect setting made by using an index register.
When an index modification is used in a sequence program, the device to be used will become the device number designated directly plus the contents of the index register.
For example, if D2Z2 has been designated the designated device is calculated as follows: D(2+3)=D5 and the content of Z2 is 3 become the designated device.

(b) There are 16 index registers, from Z0 to Z15.
Each index register can be set between -32768 and 32767.



The value -1 is stored at Z0

The data D10Z0 = D {10 + (-1)} = D9, is stored at D0

Example

A case where index modification has been performed, and the actual process device, would be as follows: (When Z0 = 20 and Z1 = -5)

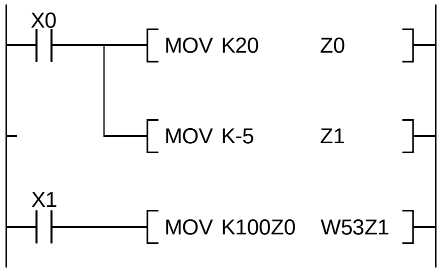
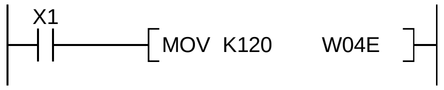
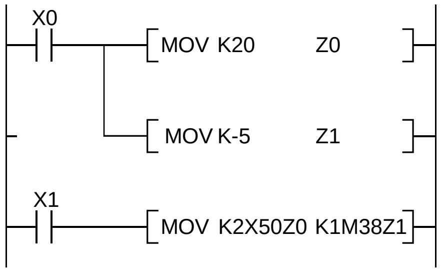
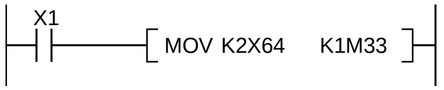
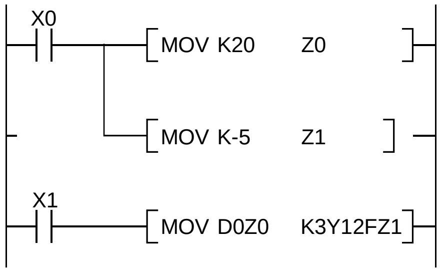
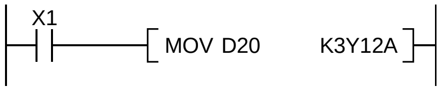
Ladder Example	Actual Process Device
 X0 MOV K20 Z0 MOV K-5 Z1 X1 MOV K100Z0 W53Z1	 X1 MOV K120 W04E Description K100Z....K (100+20) = K120 W53.......W (53-5) = W4E Hexadecimal number
 X0 MOV K20 Z0 MOV K-5 Z1 X1 MOV K2X50Z0 K1M38Z1	 X1 MOV K2X64 K1M33 Description K2X50Z....K2X (50+14) = K2X64 K20 is converted to hexadecimal K1M38......K1M (38-5) = K1M33
 X0 MOV K20 Z0 MOV K-5 Z1 X1 MOV D0Z0 K3Y12FZ1	 X1 MOV D20 K3Y12A Description D0Z.........D (0+20) = D20 K3Y12F...K3Y (12F-5) = K3Y12A Hexadecimal number

Fig. 3.7 Ladder Example and Actual Process Device

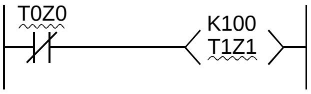
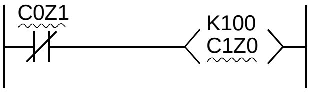
(2) Devices which can be index-modified

With the exception of the restrictions noted below, index modification can be used with devices used with contacts, coils, basic instructions, and application instructions.

(a) Devices which cannot use index modification

Device	Meaning
K, H	32-bit constant
E	Floating decimal point data
$	Character-string data
⬚ , ⬚	Bit designated for word device
FX, FY, FD	Function devices
P	Pointers used as labels
I	Interrupt pointers used as labels
Z	Index register
S	Step relay
TR	SFC transfer devices ＊1
BL	SFC block devices ＊1
T, ST	Value set for timer
C	Value set for counter

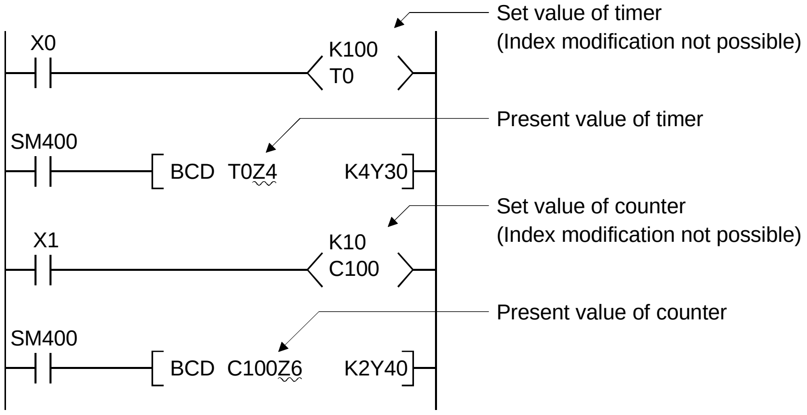
(b) Devices with limits for use with index registers

Device	Meaning	Application Example
T	• Only Z0 and Z1 can be used for timer contacts and coils	
C	• Only Z0 and Z1 can be used for counter contacts and coils	

REMARKS

1) ＊1: SFC transfer devices and SFC block devices are devices for SFC use.
Refer to the QCPU (Q mode)/QnACPU Programming Manual (SFC) for information on how to use these devices.

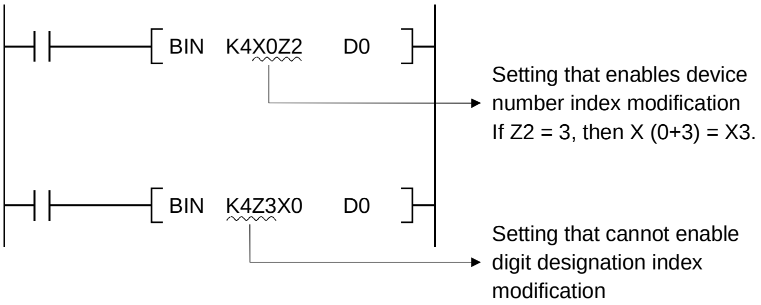
2) For timer and counter present values, there are no limits on index register numbers used.



(c) Other

1) Bit data

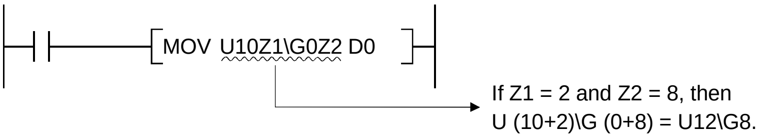
Device numbers can be index modified when performing digit designation.
However, index modification is not possible by digit designation.



Setting that enables device number index modification
If Z2 = 3, then X (0+3) = X3.

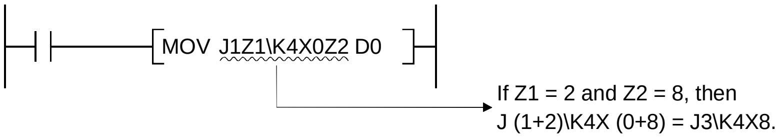
Setting that cannot enable digit designation index modification

2) Both I/O numbers and buffer memories can be index modified with special function module devices.



If Z1 = 2 and Z2 = 8, then
U (10+2)\G (0+8) = U12\G8.

3) Both network numbers and device numbers can be index modified with link direct devices.



If Z1 = 2 and Z2 = 8, then
J (1+2)\K4X (0+8) = J3\K4X8.

REMARKS

1) ✻1: Refer to the User's Manual (Functions Explanation, Programming Fundamentals) of the used CPU module or QnACPU Programming Manual (Fundamentals) for special function module device.

2) ✻2: Refer to the User's Manual (Functions Explanation, Programming Fundamentals) of the used CPU module or QnACPU Programming Manual (Fundamentals) for link direct devices.

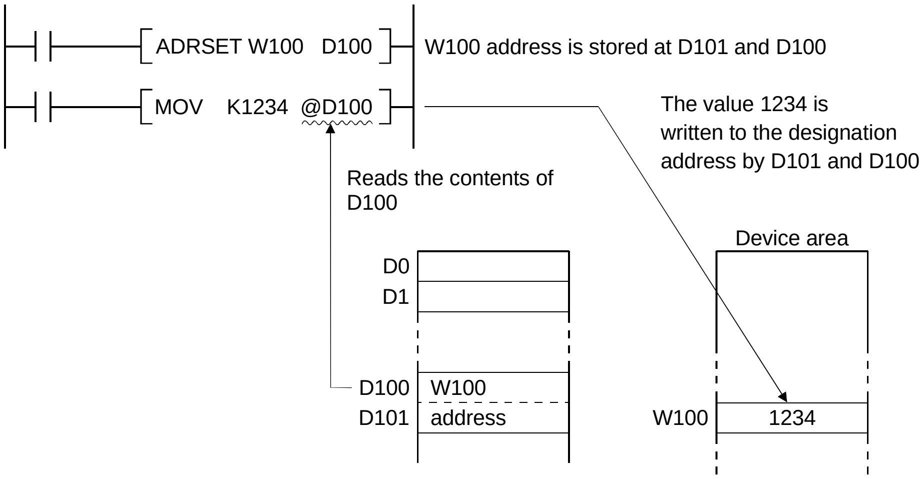
3.4 Indirect Designation

(1) Indirect Designation

(a) Indirect designation is a way of using a word device to designate a device address that will be used in a sequence program.
This method can be used when the index register is insufficient.

(b) The device which designates the designated device address is designated by "@+(word device number)".
For example, designation of @D100 will make the contents of D100 D101 the device address.

(c) The address of the device performing indirect designation can be confirmed with the ADRSET instruction.


ADRSET W100 D100
W100 address is stored at D101 and D100
MOV K1234 @D100
The value 1234 is written to the designation address by D101 and D100
Reads the contents of D100
Device area
D0
D1
D100
D101
W100
address
W100
1234

(2) Devices Capable of Indirect Designation

The CPU module devices that can be designated indirectly is shown in Table 3.3.

Table 3.3 List of Devices Capable of Indirect Designation

Device Type		Capable/Incapable of Indirect Designation	Example of Indirect Designation
Internal user devices	Bit devices ∗1	Incapable	—
	Word devices ∗1	Capable	• @D100 • @D100Z2 ∗2
Link direct devices	Bit devices ∗1	Incapable	—
	Word devices ∗1	Capable ∗3	• @J1\W10 • @J1Z1\W10Z2 ∗2
Special direct devices		Capable ∗3	• @U10\G0 • @U10Z1\G0Z2 ∗2
Index register		Incapable	—
File register		Capable	• @R0, @ZR20000 • @R0Z1, @ZR20000Z1 ∗2
Nesting		Incapable	—
Pointer			—
Constants			—
Other	SFC block devices		—
	Devices below SFC		
	Network No.		
	I/O No.		

REMARKS

1) ∗1: Refer to the User's Manual (Functions Explanation, Programming Fundamentals) of the used CPU module or QnACPU Programming Manual (Fundamentals) for device names.
2) ∗2: Indicates index modification by index register
3) ∗3: The device can be designated indirectly, however the address cannot be written in the ADRSET instruction.

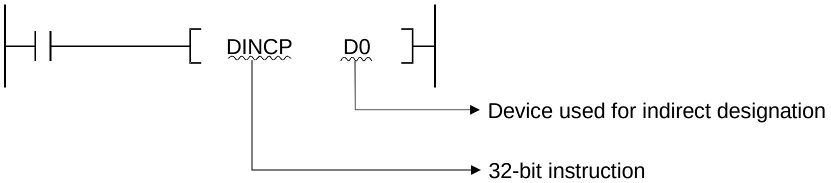
(3) Cautions

The address for indirect designation is designated using two words.

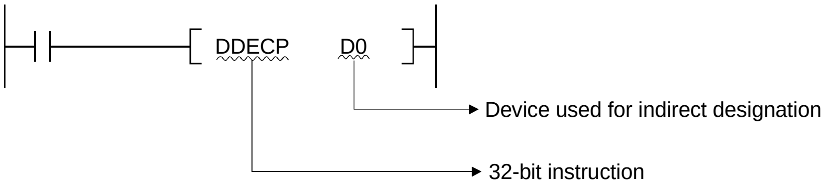
Therefore, to substitute indirect designation for index modification, the addition/subtraction of 32-bit data is required.

The following is the ladder used for the addition/subtraction of the address of the device stored in D1 and D0 for indirect designation.

[To add "1" to the address of the device for indirect designation]



[To subtract "1" from the address of the device for indirect designation]



3.5 Subset Processing

Subset processing is used to place limits on bit devices used by basic instructions and application instructions in order to increase processing speed.
However, the instruction symbol does not change.
To shorten scans, run instructions under the conditions indicated below.

(1) Conditions which each device must meet for subset processing

(a) When using word data

Device	Condition
Bit device	• Designates a bit device number in a factor of 16 • Only K4 can be designated for digit designation • Does not conduct index modification
Word device	• Internal device (File register ZR is not included)
Constants	• No limitations

(b) When using double word data

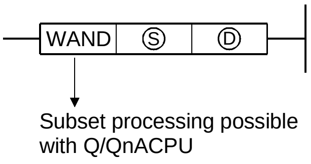
Device	Condition
Bit device	• Designates a bit device number in a factor of 32 • Only K8 can be designated for digit designation • Does not conduct index modification
Word device	• Internal device (File register ZR is not included)
Constants	• No limitations

(2) Instructions for which subset processing can be used

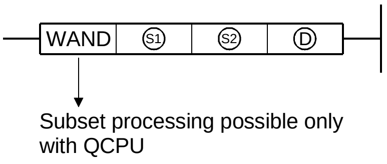
Types of Instructions	Instruction Symbols
Comparison instructions	• =, < >, <, <=, >, >=, D=, D< >, D<, D<=, D>, D>=
Basic arithmetic operations (addition, subtraction, multiplication, and division)	• +, -, ＊, /, INC, DEC, D+, D-, D＊, D/, DINC, DDEC • B+, B-, B＊, B/
Data conversion instructions	• BCD, BIN, DBCD, DBIN
Data transfer instruction	• MOV, DMOV, CML, DCML, XCH, DXCH • FMOV, BMOV, EMOV (with QCPU only)
Program branch instruction	• CJ, SCJ, JMP
Logic operations ＊	• WAND, DAND, WOR, DOR, WXOR, DXOR, WXNR, DXNR
Rotation instruction	• RCL, DRCL, RCR, DRCR, ROL, DROL, ROR, DROR
Shift instructions	• SFL, DSFL, SFR, DSFR
Data processing instructions	• SUM, SEG
Structured program instructions	• FOR, CALL

REMARK

1) ＊: It is only QCPU that can use three devices to conduct subset processing of the logic operation instructions WAND, DAND, WOR, DOR, WXOR, DXOR, WXNR, or DXNR.



Subset processing possible with Q/QnACPU



Subset processing possible only with QCPU

3.6 Cautions on Programming (Operation Errors)

Operation errors are returned in the following cases when executing basic instructions and application instructions with CPU module:

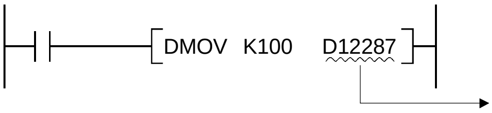
- An error listed on the explanatory page for the individual instruction occurred.
- No intelligent function module or special function module is installed at the designated I/O No. position when using the buffer register.
- The relevant network does not exist when using a link device.
- No network module is installed at the designated I/O No. when using a link device.

POINT

(1) When a file register setting has been made but no memory card has been installed, or when no file register setting has been made, no error will be returned even if an attempt is made to write to the file register.
However, "FFFFH" will be stored if an attempt is made to read from the file register at which this write operation was attempted.

(1) Device range check
Device range checks for the devices used by basic instructions and application instructions in CPU module are as indicated below:

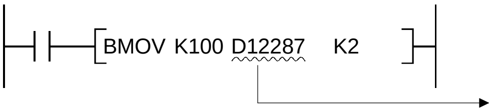
(a) No device range check is made for instructions dealing with fixed-length devices (MOV, DMOV, etc.).
In cases where the corresponding device range is exceeded, data is written to other devices. ✻
For example, in a case where the data register has been allocated 12 k points, there will be no error even if it exceeds D12287.



D12287 and D12288 have been indicated here, but because D12288 does not exist, the contents of some other device will be destroyed.

Device range checks are not conducted also in cases where index modification is being performed.

(b) Device range checks are conducted for instructions dealing with variable-length devices (BMOV, FMOV, and others which designate transfer numbers).
In cases where the corresponding device range has been exceeded, an operation error will be returned.
For example, in a case where the data register has been allocated 12 k points, there will be an error if it exceeds D12287.

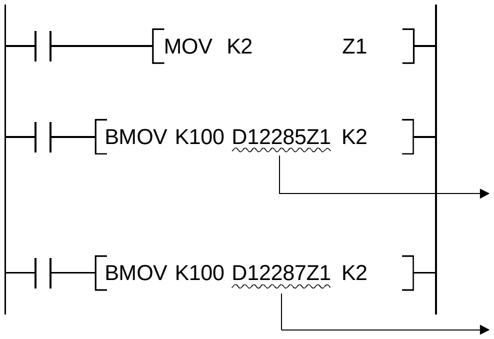


D12287 and D12288 have been indicated here, but because D12288 does not exist, an operation error is returned.

REMARK

1) ✻: See section 3.4 (3) for the internal user device allocation order.

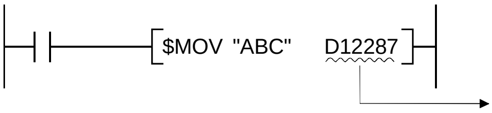
Device range checks are also conducted when index modification is performed.
However, if index modification has been conducted, there will be no error returned if the initial device number exceeds the relevant device range.



D12287 and D12288 have been indicated here, but because D12288 does not exist, an operation error is returned.

Because the initial device number is D12289 and that exceeds the device range, the initial device number is made W0, the operation is conducted, and no error is returned.

(c) Because all character string data is of variable length, device range checks are performed.
In cases where the corresponding device range has been exceeded, an operation error will be returned.
For example, in a case where the data register has been allocated 12 k points, there will be an error if it exceeds D12287.



D12287 and D12288 have been indicated here, but because D12288 does not exist, an operation error is returned.

Note that an operation error does not occur even if the head device number exceeds the device range as the result of index modification.

(d) Device range checks are conducted when index modification is performed by direct access output (DY).

(2) Device data check
Device data checks for the devices used by basic instructions and application instructions in CPU module are as indicated below:

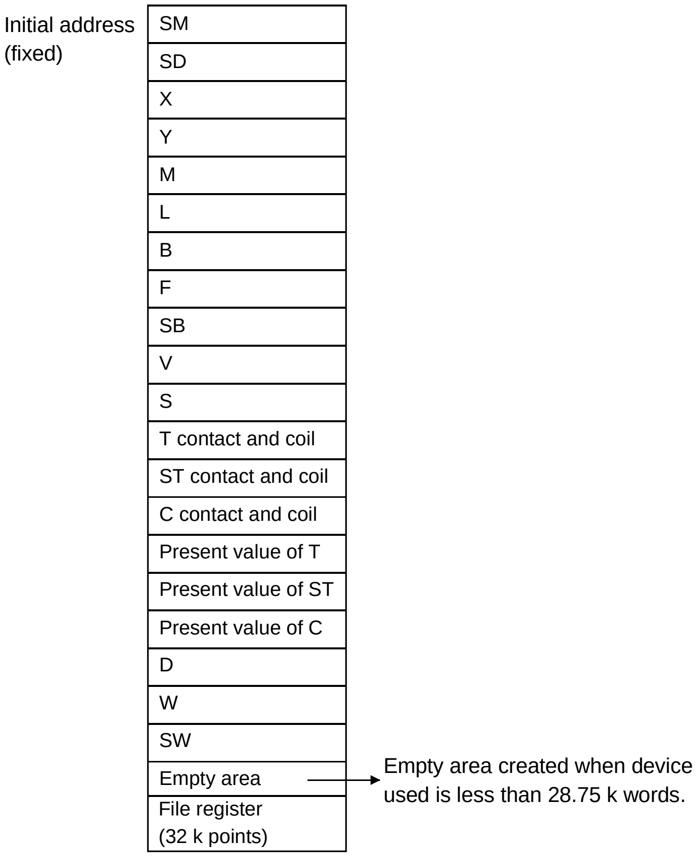
(a) When using BIN data
- No error is returned even if the operation results in overflow or underflow.
  The carry flag does not go on at such times, either.

(b) When using BCD data
1) Each digit is check for BCD value (0/ to 9).
   An operation error is returned if individual digits are outside the 0 to 9 (A to F) range.
2) No error is returned even if the operation results in overflow or underflow.
   The carry flag does not go on at such times, either.

(c) When using floating decimal point data
Operation errors are returned in the following cases:
- When value of floating decimal point data is 0
- When the absolute value of the floating decimal point data is 1.0 × 2^{-127} or lower
- When absolute value of floating decimal point data is 1.0 × 2^{128} or higher

(d) When using character string data
No data check is conducted.

(3) If internal user device allocation is changed by parameter device allocation, such allocations are made in the device order indicated below:
If the allocation of the device used is less than 28.75 k words, the area following the device used will be empty.



REMARK

1) Refer to the User's Manual (Functions Explanation, Programming Fundamentals) of the used CPU module or the QnACPU Programming Manual (Fundamentals), for how to change the internal user device allocation.

3.7 Conditions for Execution of Instructions

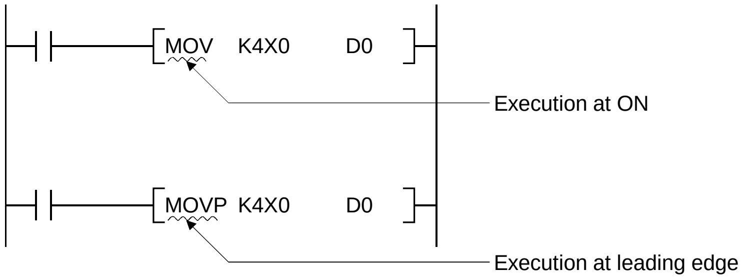
The following four types of execution conditions exist for the execution of CPU module sequence instructions, basic instructions, and application instructions:

- Non-conditional execution.................. Instructions executed without regard to the ON/OFF status of the device
  Example: LD X0, OUT Y10
- Executed at ON.................................. Instructions executed while input condition is ON
  Example: MOV instruction, FROM instruction
- Executed at leading edge .... Instructions executed only at the leading edge of the input condition (when it goes from OFF to ON)
  Example: PLS instruction, MOVP instruction
- Executed at trailing edge .... Instructions executed only at the trailing edge of the input condition (when it goes from ON to OFF)
  Example: PLF instruction

For coil or equivalent basic instructions or application instructions, where the same instruction can be designated for either execution at ON or leading edge execution, a "P" is added after the instruction name to specify the condition for execution.

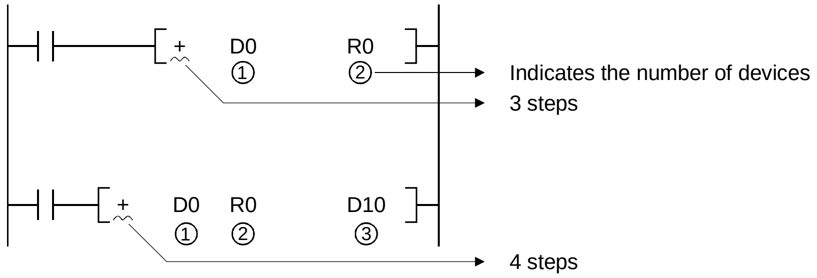
- Instruction to be executed at ON Instruction name
- Instruction to be executed at leading edge Instruction name + P

Execution at ON and execution at leading edge for the MOV instruction are designated as follow:



3.8 Counting Step Number

The number of steps in CPU module sequence instructions, basic instructions, and application instructions differs depending on whether indirect setting of the device used is possible or not.
The basic number of steps for basic instructions and application instructions is calculated by adding the device number and 1.
For example, the "+ instruction" would be calculated as follows:



(1) Conditions for increasing the number of steps
The number of steps is increased over the number of basic steps in cases where a device is used that is designated indirectly or for which the number of steps is increased.

(a) When device is designated indirectly
In cases where indirect designation is done by @☐, the number of steps is increased 1 step over the number of basic steps.
For example, when a 3-step MOV instruction is designated indirectly (example: MOV K4X0 @D0), one step is added and the instruction becomes 4 steps.

(b) Devices where number of steps increases

Devices Where Number of Steps Increases	Added Steps	Example
Intelligent function module device/special function module device	1	MOV U4\G10 D0
Link direct devices		MOV J3\B20 D0
Serial number access file registers		MOV ZR123 D0
32-bit constants		DMOV K123 D0
Real number constant		EMOV E0.1 D0
Character string constant	For even numbers: (number of characters)/2 For odd numbers: (number of characters + 1)/2	$MOV "123" D0

(c) In cases where the conditions described in (a) and (b) above overlap, the number of steps becomes a culmination of the two.
For example, if MOV U1\G10 ZR123 has been designated, 1 step is added for buffer register designation and 1 step is added for serial number access file register designation, making a total of 2 steps added.

3.9 Operation when OUT, SET/RST, or PLS/PLF Instructions Use the Same Device

The following describes the operation for executing multiple instructions of OUT, SET/RST, or PLS/PLF that use the same device in one scan.

(1) OUT instructions using the same device

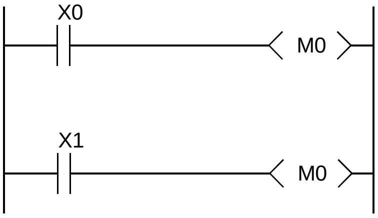
Do not program more than one OUT instruction using the same device in one scan.

If the OUT instructions using the same device are programmed in one scan, the specified device will turn ON or OFF every time the OUT instruction is executed, depending on the operation result of the program up to the relevant OUT instruction.

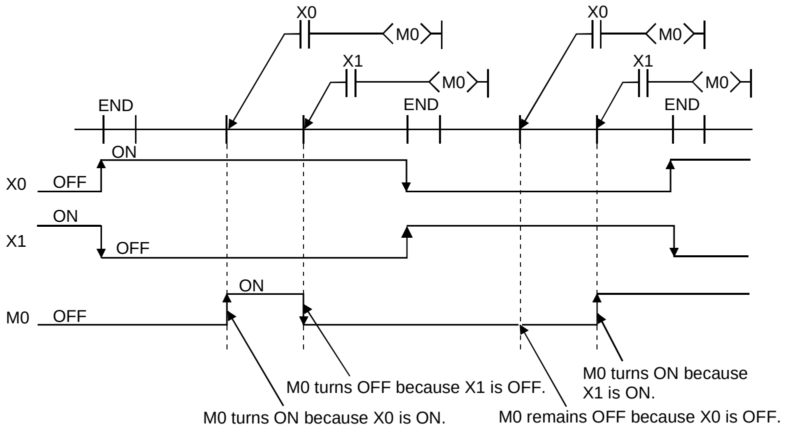
Since turning ON or OFF of the device is determined when each OUT instruction is executed, the device may turn ON and OFF repeatedly during one scan.

The following diagram shows an example of a circuit that turns the same internal relay (M0) with inputs X0 and X1 ON and OFF.

[Circuit]



[Timing Chart]


X0
M0
X1
M0
END
END
END
X0
M0
X1
M0
X0
OFF
ON
X1
ON
OFF
M0
OFF
ON
M0 turns OFF because X1 is OFF.
M0 turns ON because X0 is ON.
M0 turns ON because X1 is ON.
M0 remains OFF because X0 is OFF.

With the refresh type CPU module, when the output (Y) is specified by the OUT instruction, the ON/OFF status of the last OUT instruction of the scan will be output.

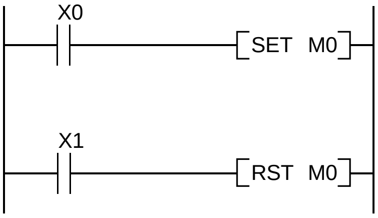
(2) SET/RST instructions using the same device

(a) The SET instruction turns ON the specified device when the SET command is ON and does not do anything when the SET command is OFF.
For this reason, when two or more SET instructions use the same device in one scan, the specified device will be ON if any one of the SET commands is ON.

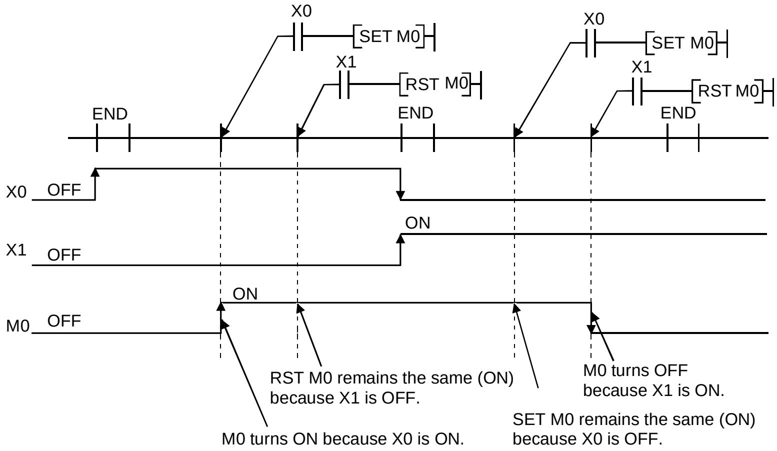
(b) The RST instruction turns OFF the specified device when the RST command is ON and does not anything when the RST command is OFF.
For this reason, when two or more RST instructions use the same device in one scan, the specified device will be OFF if any one of the RST commands is ON.

(c) When the SET instruction and RST instruction using the same device are programmed in one scan, the SET instruction turns ON the specified device when the SET command is ON and the RST instruction turns OFF the specified device when the RST command is ON.
When both the SET and RST commands are OFF, the ON/OFF status of the specified device will not be changed.

[Circuit]



[Timing Chart]


X0
SET M0
X1
RST M0
X0
SET M0
X1
RST M0
END
END
END
X0
OFF
X1
OFF
ON
M0
OFF
ON
RST M0 remains the same (ON) because X1 is OFF.
M0 turns OFF because X1 is ON.
SET M0 remains the same (ON) because X0 is OFF.
M0 turns ON because X0 is ON.

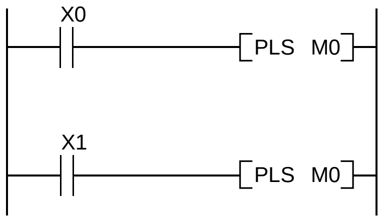
(3) PLS instructions using the same device

The PLS instruction turns ON the specified device when the PLS command turns ON from OFF. It turns OFF the specified device at any other time (OFF → OFF, ON → ON, and ON → OFF).

When two or more PLS instructions using the same device are programmed in one scan, each PLS instruction turns ON the specified device when the corresponding PLS command turns ON from OFF and it turns OFF the specified device at any other time.

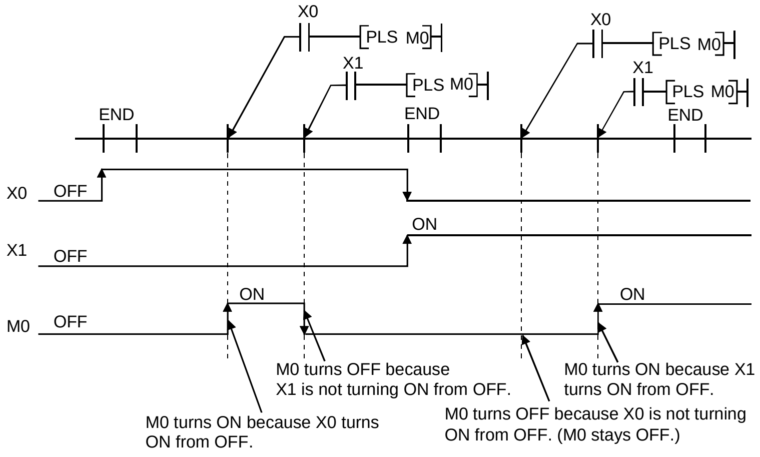
For this reason, when two or more PLS instructions using the same device are programmed in one scan, the device that has been turned ON by the PLS command may not turn ON again throughout the scan.

[Circuit]

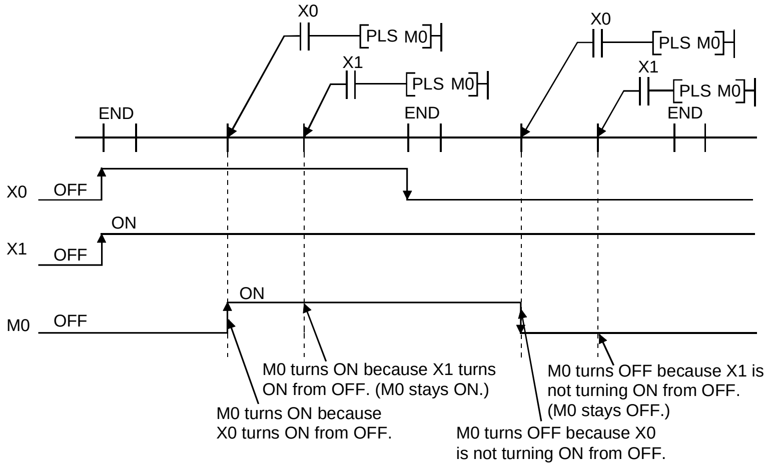


[Timing Chart]

- The ON/OFF timing of the X0 and X1 is different. (The specified device does not turn ON throughout the scan.)


X0
PLS M0
X1
PLS M0
X0
PLS M0
X1
PLS M0
END
END
END
X0
OFF
X1
OFF
ON
M0
OFF
ON
ON
M0 turns OFF because X1 is not turning ON from OFF.
M0 turns ON because X1 turns ON from OFF.
M0 turns ON because X0 turns ON from OFF.
M0 turns OFF because X0 is not turning ON from OFF. (M0 stays OFF.)

- The X0 and X1 turn ON from OFF at the same time.


X0
PLS M0
X1
PLS M0
X0
PLS M0
X1
PLS M0
END
END
END
X0
OFF
ON
X1
OFF
ON
M0
OFF
M0 turns ON because X1 turns ON from OFF. (M0 stays ON.)
M0 turns OFF because X1 is not turning ON from OFF. (M0 stays OFF.)
M0 turns ON because X0 turns ON from OFF.
M0 turns OFF because X0 is not turning ON from OFF.

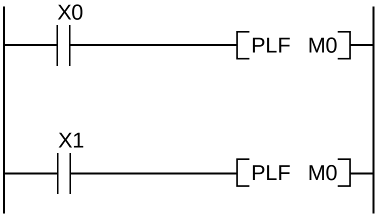
(4) PLF instructions using the same device

The PLF instruction turns ON the specified device when the PLF command turns ON from OFF. It turns OFF the specified device at any other time (OFF → OFF, OFF → ON, and ON → ON).

When two or more PLF instructions using the same device are programmed in one scan, each PLF instruction turns OFF the specified device when the corresponding PLF command turns OFF from ON and it turns OFF the specified device at any other time.

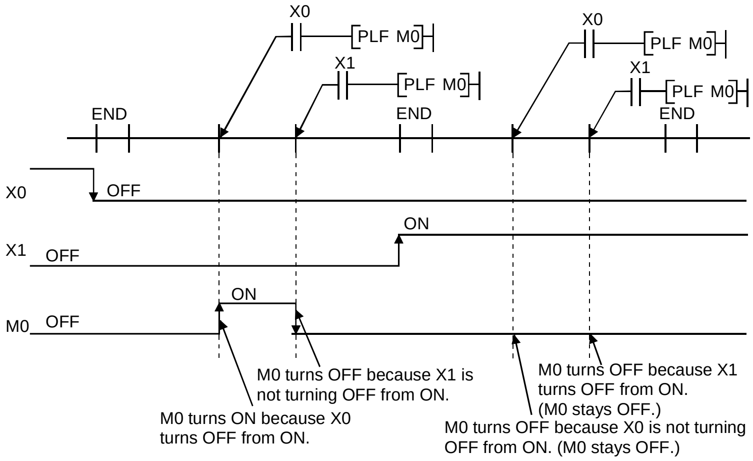
For this reason, when two or more PLF instructions using the same device are programmed in one scan, the device that has been turned ON by the PLF command may not turn ON again throughout the scan.

[Circuit]



[Timing Chart]

- The ON/OFF timing of the X0 and X1 is different. (The specified device does not turn ON throughout the scan.)



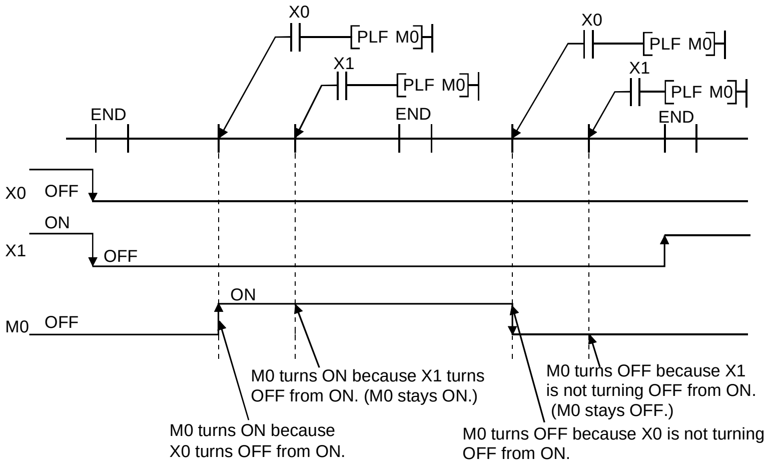
M0 turns OFF because X1 is not turning OFF from ON.

M0 turns ON because X0 turns OFF from ON.

M0 turns OFF because X1 turns OFF from ON. (M0 stays OFF.)

M0 turns OFF because X0 is not turning OFF from ON. (M0 stays OFF.)

- The X0 and X1 turn OFF from ON at the same time.



M0 turns ON because X1 turns OFF from ON. (M0 stays ON.)

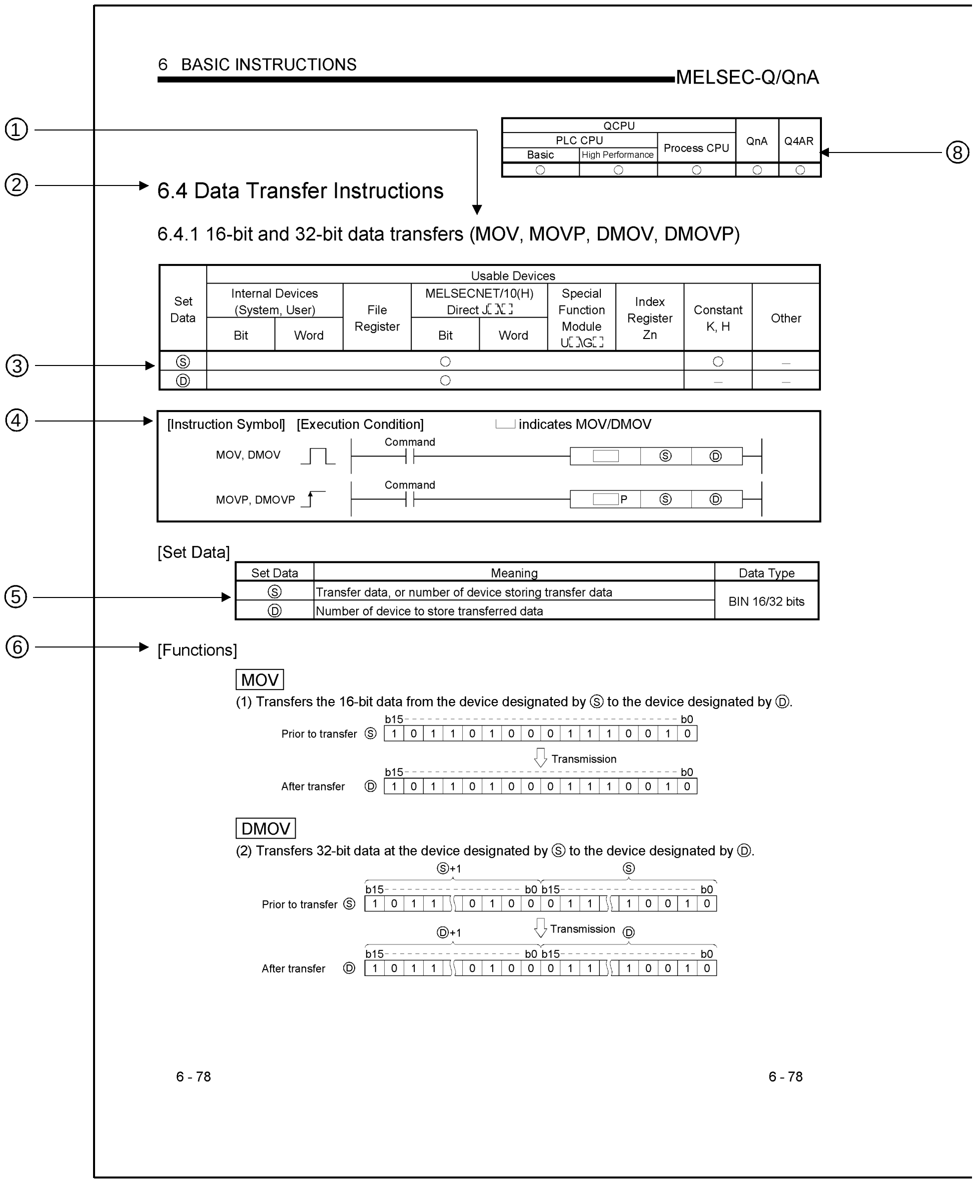
M0 turns ON because X0 turns OFF from ON.

M0 turns OFF because X1 is not turning OFF from ON. (M0 stays OFF.)

M0 turns OFF because X0 is not turning OFF from ON.

4. HOW TO READ INSTRUCTIONS

The description of instructions that are contained in the following chapters are presented in the following format.



6 BASIC INSTRUCTIONS
MELSEC-Q/QnA

QCPU			QnA	Q4AR
PLC CPU		Process CPU		
Basic	High Performance			
○	○	○	○	○

6.4 Data Transfer Instructions

6.4.1 16-bit and 32-bit data transfers (MOV, MOVP, DMOV, DMOVP)

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J⌷\⌷		Special Function Module U⌷\G⌷	Index Register Zn	Constant K, H	Other
	Bit	Word		Bit	Word				
Ⓢ	○							○	–
Ⓓ	○							–	–

[Instruction Symbol] [Execution Condition] ⌴ indicates MOV/DMOV

MOV, DMOV — Command — ⌴ Ⓢ Ⓓ

MOVP, DMOVP — Command — ⌴P Ⓢ Ⓓ

[Set Data]

Set Data	Meaning	Data Type
Ⓢ	Transfer data, or number of device storing transfer data	BIN 16/32 bits
Ⓓ	Number of device to store transferred data	

[Functions]

MOV

(1) Transfers the 16-bit data from the device designated by Ⓢ to the device designated by Ⓓ.

		b15															b0
Prior to transfer	Ⓢ	1	0	1	1	0	1	0	0	0	1	1	1	0	0	1	0
After transfer	Ⓓ	1	0	1	1	0	1	0	0	0	1	1	1	0	0	1	0

Transmission

DMOV

(2) Transfers 32-bit data at the device designated by Ⓢ to the device designated by Ⓓ.

Ⓢ+1: b15 1 0 1 1 … 0 1 0 0 b0 ; Ⓢ: b15 0 1 1 … 1 0 0 1 0 b0 — Prior to transfer Ⓢ

Transmission

Ⓓ+1: b15 1 0 1 1 … 0 1 0 0 b0 ; Ⓓ: b15 0 1 1 … 1 0 0 1 0 b0 — After transfer Ⓓ

6 - 78 6 - 78

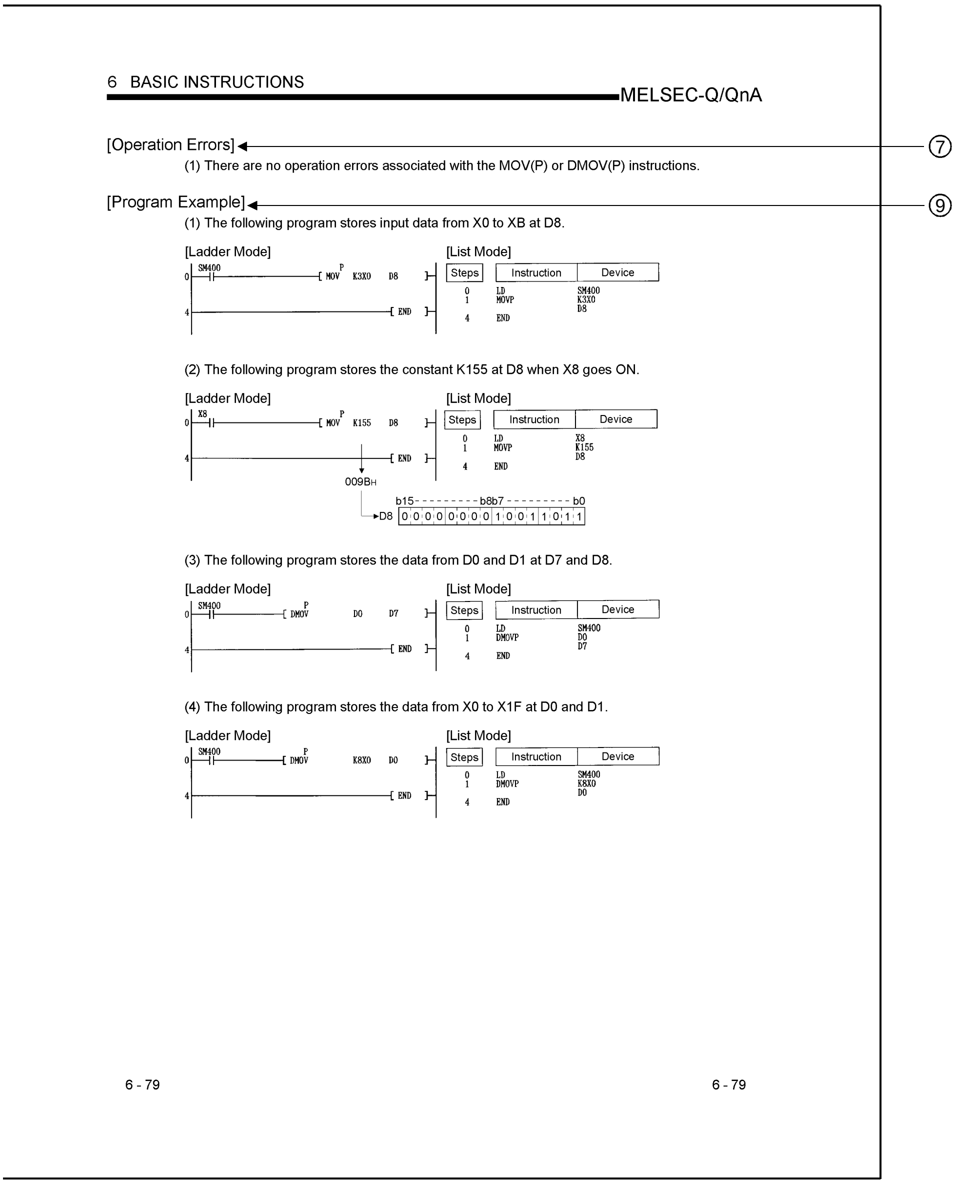
① Code used to write instruction (instruction symbol).

② Section number and general category of instructions being discussed.

③ Devices which can be used by the instruction in question are indicated with circle.
The types of devices that can be used are as indicated below:

Device Type	Internal Devices (System, User)		File Register	MELSECNET/10(H) ∗3 Direct J⌷\⌷		Special Function Module U⌷\G⌷	Index Register Zn	Constant ∗1	Other ∗1
	Bit	Word		Bit	Word				
Usable ∗4 devices	X, Y, M, L, SM, F, V, B, SB, FX, FY ∗2	T, ST, C, ∗5 D, W, SD, SW, FD, @⌷	R, ZR	J⌷\X J⌷\Y J⌷\B J⌷\SB	J⌷\W J⌷\SW	U⌷\G⌷	Z	Decimal constants Hexadecimal constants Real number constant Character string constant	P, I, J, U, DX, DY, N, BL, TR, BL\S, V

∗1: Devices which can be set are recorded in the "Constant" and the "Other" columns.
∗2: FX and FY can be used only for bit data, and FD only for word data.
∗3: Usable with the MELSECNET/H, and MELSECNET/10.



6 BASIC INSTRUCTIONS MELSEC-Q/QnA

[Operation Errors] ⑦

(1) There are no operation errors associated with the MOV(P) or DMOV(P) instructions.

[Program Example] ⑨

(1) The following program stores input data from X0 to XB at D8.

[Ladder Mode]

```
   SM400         P
0 ─┤├────────[ MOV  K3X0   D8  ]─
4 ──────────────────────[ END  ]─
```

[List Mode]

Steps	Instruction	Device
0	LD	SM400
1	MOVP	K3X0 D8
4	END	

(2) The following program stores the constant K155 at D8 when X8 goes ON.

[Ladder Mode]

```
   X8            P
0 ─┤├────────[ MOV  K155   D8  ]─
4 ──────────────────────[ END  ]─
                  009BH
        b15---------b8b7---------b0
     ─►D8 0 0 0 0 0 0 0 0 1 0 0 1 1 0 1 1
```

[List Mode]

Steps	Instruction	Device
0	LD	X8
1	MOVP	K155 D8
4	END	

(3) The following program stores the data from D0 and D1 at D7 and D8.

[Ladder Mode]

```
   SM400      P
0 ─┤├─────[ DMOV     D0     D7  ]─
4 ──────────────────────[ END  ]─
```

[List Mode]

Steps	Instruction	Device
0	LD	SM400
1	DMOVP	D0 D7
4	END	

(4) The following program stores the data from X0 to X1F at D0 and D1.

[Ladder Mode]

```
   SM400      P
0 ─┤├─────[ DMOV     K8X0   D0  ]─
4 ──────────────────────[ END  ]─
```

[List Mode]

Steps	Instruction	Device
0	LD	SM400
1	DMOVP	K8X0 D0
4	END	

6 - 79 6 - 79

∗4: Refer to the User's Manual (Functions Explanation, Programming Fundamentals) of the used CPU or QnA Programming Manual for details of each device.

∗5: When T, ST and C are used for other than the instructions below, only word data can be used. (Bit data cannot be used .)
[Instructions that can be used with bit data]
LD, LDI, AND, ANI, OR, ORI, LDP, LDF, ANDP, ANDF, ORP, ORF, OUT, RST

④ Indicates ladder mode expressions and execution conditions for instructions.

Execution Condition	Non-conditional Execution	Executed while ON	Executed One Time at ON	Executed while OFF	Executed One Time at OFF
Code recorded on description page	No symbol recorded	_┌┐_	_↑‾	‾└┘‾	‾↓_

⑤ Discusses the data set for each instruction and the data type.

Data Type	Meaning
Bit	Bit data or first number in bit data
BIN 16 bits	BIN 16-bit data or first number in word device
BIN 32 bits	BIN 32-bit data or first number in double word device
BCD 4 digits	4-digit BCD data
BCD 8 digits	8-digit BCD data
Real number	Floating decimal point data
Character string	Character string data
Device name	Device name data

⑥ Indicates the function of the instruction.

⑦ Indicates conditions under which error is returned, and error number.
See Section 3.6 for errors not included here.

⑧ Indicates whether the instruction can be used with each CPU module type.
○: Can be used
△: Can be used with restrictions (function version, software version)
×: Cannot be used

⑨ Indicates both ladder and list for simple program example.
Also indicates the types of individual devices used when the program is executed.

5. SEQUENCE INSTRUCTIONS

Sequence instructions include instructions for relay control ladders and the like. They are divided into the following categories:

Instruction	Meaning	Reference
Contact instruction	Operation start, series connection, parallel connection	Chapter 5.1
Connection Instruction	Ladder block connection, creation of pulses from operation results, store/read operation results	Chapter 5.2
Output instruction	Bit device output, pulse output, output reversal	Chapter 5.3
Shift instruction	Bit device shift	Chapter 5.4
Master control instruction	Master control	Chapter 5.5
Termination instruction	Program termination	Chapter 5.6
Other instructions	Program stop, instructions such as no operation which do not fit in the above categories	Chapter 5.7

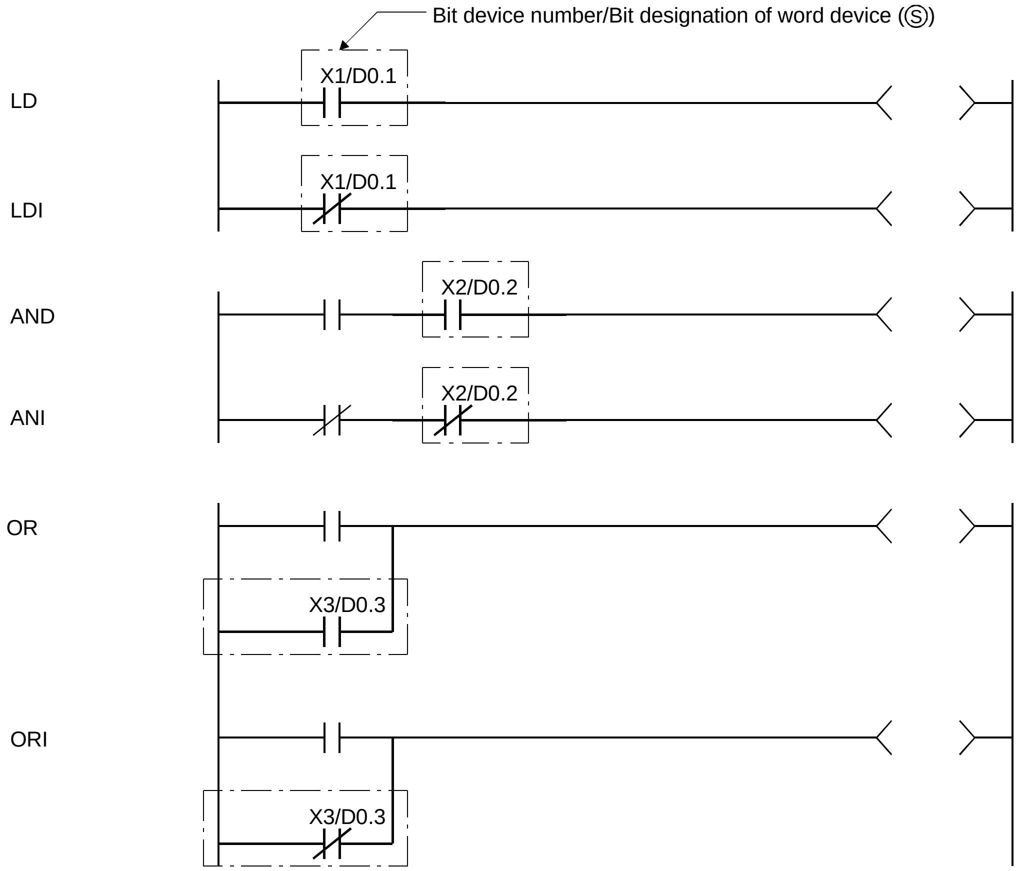
QCPU			QnA	Q4AR
PLC CPU		Process CPU		
Basic	High Performance			
○	○	○	○	○

5.1 Contact Instructions

5.1.1 Operation start, series connection, parallel connection (LD, LDI, AND, ANI, OR, ORI)

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant K, H	Other
	Bit	Word		Bit	Word				DX, BL
Ⓢ	○						—		○

[Instruction Symbol] [Execution Condition]



[Set Data]

Set Data	Meaning	Data Type
Ⓢ	Devices used as connections	Bit

[Functions]

LD, LDI

(1) LD is the A contact operation start instruction, and LDI is the B contact operation start instruction. They read ON/OFF information from the designated device (if a word device bit has been designated, this becomes the 1/0 status of the designated bit), and use that as an operation result.

AND, ANI

(1) AND is the A contact series connection instruction, and ANI is the B contact series connection instruction. They read the ON/OFF data of the designated bit device (if a bit designation has been made for a word device, the 1/0 status of the designated bit is read), perform an AND operation on that data and the operation result to that point, and take this value as the operation result.

(2) There are no restrictions on the use of AND or ANI, but the following applies with a peripheral device used in the ladder mode:

(a) Write...........When AND and ANI are connected in series, a ladder with up to 21 stages can be generated.

(b) Read ....When AND and ANI are connected in series, a ladder with up to 24 stages can be displayed.
If the number exceeds 24 stages, up to 24 will be displayed.

OR, ORI

(1) OR is the A contact single parallel connection instruction, and ORI is the B contact single parallel connection instruction. They read ON/OFF information from the designated device (if a word device bit has been designated, this becomes the 1/0 status of the designated bit), and perform an OR operation with the operation results to that point, and use the resulting value as the operation result.

(2) There are no limits on the use of OR or ORI, but the following applies with a peripheral device used in the ladder mode.

(a) Write................OR and ORI can be used to create connections of up to 23 ladders.

(b) Read................Up to 23 ladders connected with OR or ORI can be displayed.
The 24th or subsequent ladders cannot be displayed properly.

REMARK

Word device bit designations are made in hexadecimal
Bit b11 of D0 would be D0.0B.
See Section 3.2.1 for more information on word device bit designation.

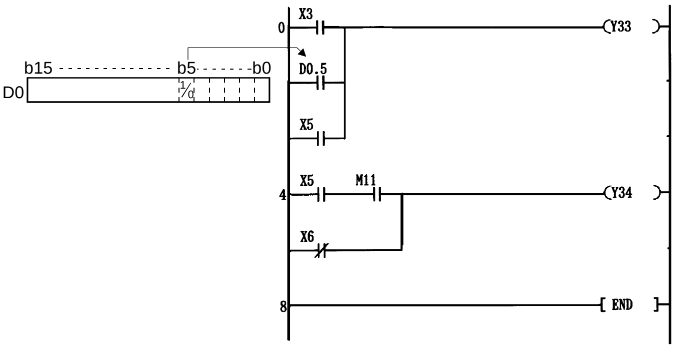
[Operation Errors]

(1) There are no operation errors with LD, LDI, AND, ANI, OR, or ORI instructions.

[Program Example]

(1) A program using LD, AND, OR, and ORI instructions.

[Ladder Mode]

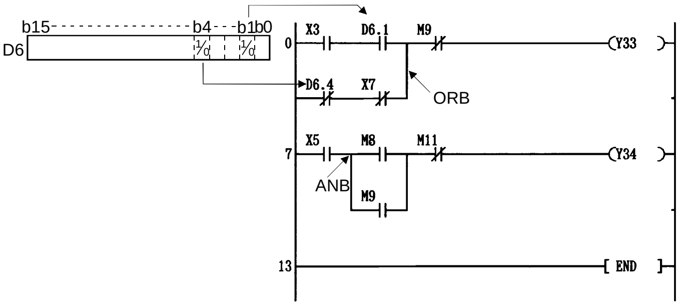


[List Mode]

Steps	Instruction	Device
0	LD	X3
1	OR	D0.5 ···Word device bit designation
2	OR	X5
3	OUT	Y33
4	LD	X5
5	AND	M11
6	ORI	X6
7	OUT	Y34
8	END	

(2) A program linking contact points established through the use of ANB and ORB instructions.

[Ladder Mode]

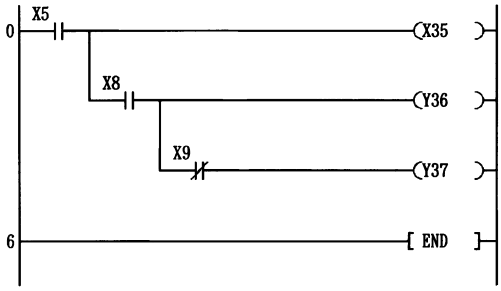


[List Mode]

Steps	Instruction	Device
0	LD	X3
1	AND	D6.1 }···Word device bit designation
2	LDI	D6.4
3	ANI	X7
4	ORB	
5	ANI	M9
6	OUT	Y33
7	LD	X5
8	LD	M8
9	OR	M9
10	ANB	
11	ANI	M11
12	OUT	Y34
13	END	

(3) A parallel program with OUT instruction

[Ladder Mode]



[List Mode]

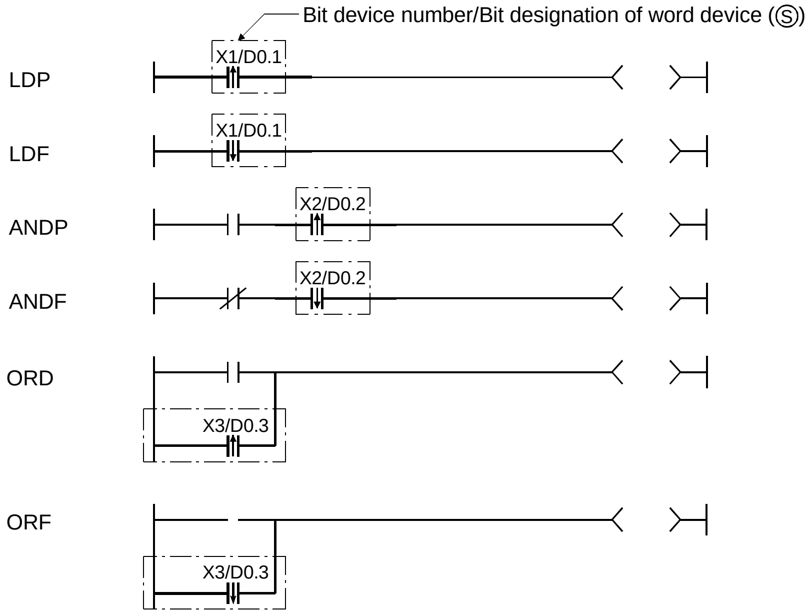
Steps	Instruction	Device
0	LD	X5
1	OUT	X35
2	AND	X8
3	OUT	Y36
4	ANI	X9
5	OUT	Y37
6	END	

QCPU			QnA	Q4AR
PLC CPU		Process CPU		
Basic	High Performance			
○	○	○	○	○

5.1.2 Pulse operation start, pulse series connection, pulse parallel connection (LDP, LDF, ANDP, ANDF, ORP, ORF)

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant K, H	Other
	Bit	Word		Bit	Word				DX
Ⓢ	○						–		○

[Instruction Symbol] [Execution Condition]



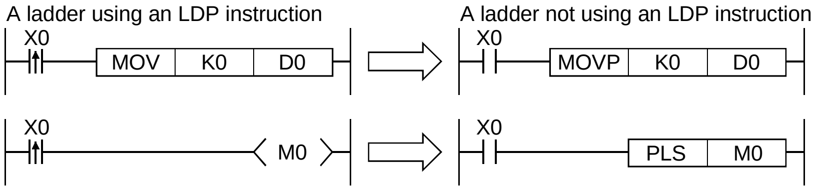
[Set Data]

Set Data	Meaning	Data Type
Ⓢ	Devices used as contacts	Bit

[Functions]

LDP, LDF

(1) LDP is the leading edge pulse operation start instruction, and is ON only at the leading edge of the designated bit device (when it goes from OFF to ON).
If a word device has been designated, it is ON only when the designated bit changes from 0 to 1.
In cases where there is only an LDP instruction, it acts identically to instructions for the creation of a pulse that are executed during ON(⁝⁝⁝ P).


A ladder using an LDP instruction
A ladder not using an LDP instruction
X0
MOV
K0
D0
X0
MOVP
K0
D0
X0
M0
X0
PLS
M0

(2) LDF is the trailing edge pulse operation start instruction, and is ON only at the trailing edge of the designated bit device (when it goes from ON to OFF).
If a word device has been designated, it is ON only when the designated bit changes from 1 to 0.

ANDP, ANDF

(1) ANDP is a leading edge pulse series connection instruction, and ANDF is a trailing edge pulse series connection instruction. They perform an AND operation with the operation result to that point, and take the resulting value as the operation result.
The ON/OFF data used by ANDP and ANDF are indicated in the table below:

Devices Designated by ANDP		ANDP State	Devices Designated by ANDF		ANDF State
Bit Device	Word Device Bit Designation		Bit Device	Word Device Bit Designation	
OFF → ON	0 → 1	ON	OFF → ON	0 → 1	OFF
OFF	0	OFF	OFF	0	
ON	1		ON	1	
ON → OFF	1 → 0		ON → OFF	1 → 0	ON

ORP, ORF

(1) ORP is a leading edge pulse parallel connection instruction, and ORF is a trailing edge pulse parallel connection instruction. They perform an OR operation with the operation result to that point, and take the resulting value as the operation result.

Devices Designated by ORP		ORP State	Devices Designated by ORF		ORF State
Bit Device	Word Device Bit Designation		Bit Device	Word Device Bit Designation	
OFF → ON	0 → 1	ON	OFF → ON	0 → 1	OFF
OFF	0	OFF	OFF	0	
ON	1		ON	1	
ON → OFF	1 → 0		ON → OFF	1 → 0	ON

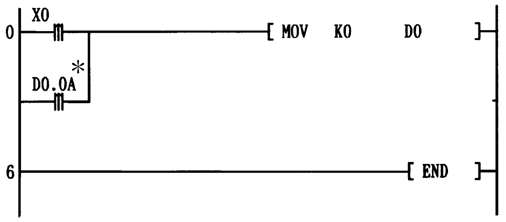
[Operation Errors]

(1) There are no operation errors with LDP, LDF, ANDP, ANDF, ORP, or ORF instructions.

[Program Example]

(1) The following program executes the MOV instruction at input X0, or at the leading edge of b10 (bit 10) of data register D0:

[Ladder Mode]



[List Mode]

Steps	Instruction	Device
0	LDP	X0
2	ORP	D0.0A
4	MOV	K0 D0
6	END	

REMARK

*: Word device bit designations are performed in hexadecimal.
Bit b10 of D0 would be D0.0A.

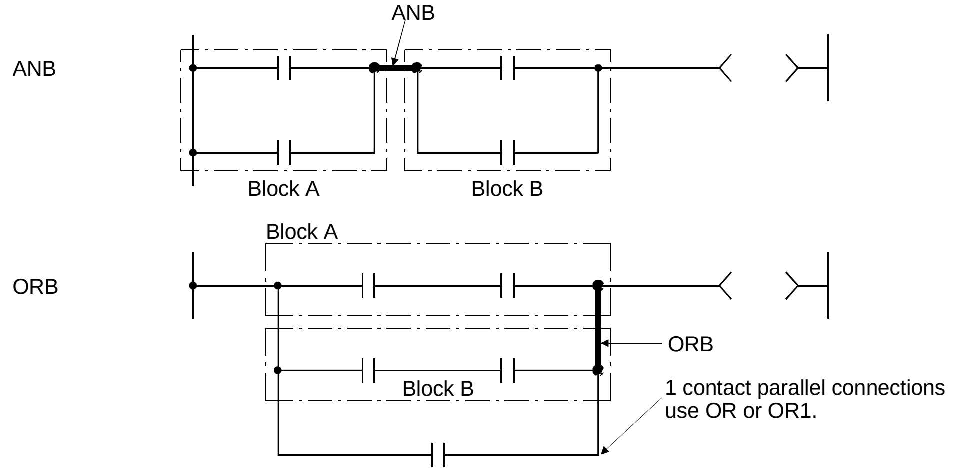
QCPU			QnA	Q4AR
PLC CPU		Process CPU		
Basic	High Performance			
○	○	○	○	○

5.2 Connection Instructions

5.2.1 Ladder block series connections and parallel connections (ANB, ORB)

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant K, H	Other
	Bit	Word		Bit	Word				
—	—								

[Instruction Symbol] [Execution Condition]



[Functions]

ANB

(1) Performs an AND operation on block A and block B, and takes the resulting value as the operation result.

(2) The symbol for ANB is not the contact symbol, but rather is the connection symbol.

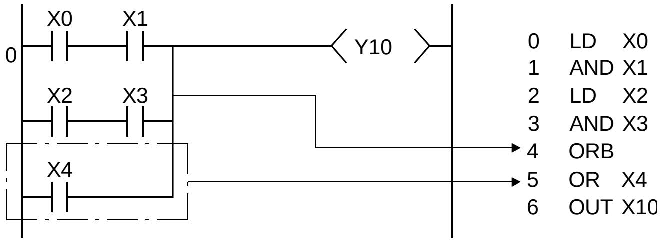
(3) When programming in the list mode, up to 15 ANB instructions (16 blocks) can be written consecutively.

ORB

(1) Conducts an OR operation on Block A and Block B, and takes the resulting value as the operation result.

(2) ORB is used to perform parallel connections for ladder blocks with two or more contacts. For ladder blocks with only one contact, use OR or ORI; there is no need for ORB in such cases.

[Ladder Mode] [List Mode]



```
0   LD   X0
1   AND  X1
2   LD   X2
3   AND  X3
4   ORB
5   OR   X4
6   OUT  X10
```

(3) The ORB symbol is not the contact symbol, but rather is the connection symbol.

(4) When programming in the list mode, it is possible to use up to 15 ORB instructions successively (16 blocks).

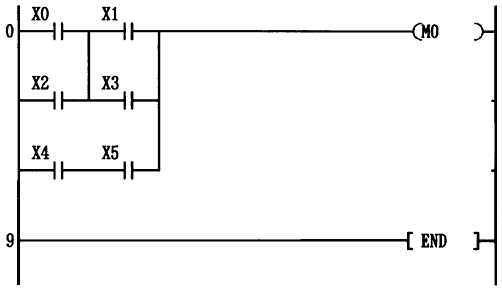
[Operation Errors]

(1) There are no operation errors associated with ANB or ORB instructions.

[Program Example]

(1) A program using ANB and ORB instructions

[Ladder Mode] [List Mode]



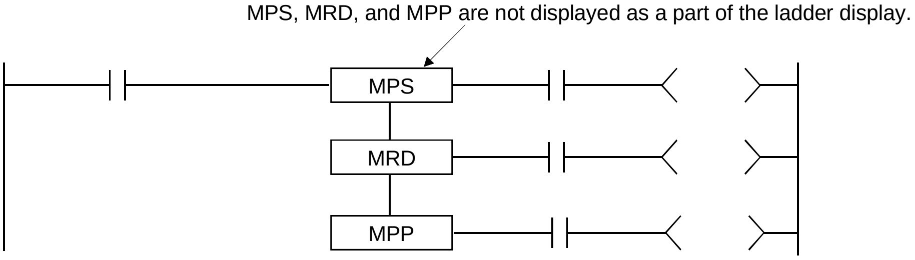
Steps	Instruction	Device
0	LD	X0
1	OR	X2
2	LD	X1
3	OR	X3
4	ANB	
5	LD	X4
6	AND	X5
7	ORB	
8	OUT	M0
9	END	

QCPU			QnA	Q4AR
PLC CPU		Process CPU		
Basic	High Performance			
○	○	○	○	○

5.2.2 Operation results push, read, pop (MPS, MRD, MPP)

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant K, H	Other
	Bit	Word		Bit	Word				
—	—								

[Instruction Symbol] [Execution Condition]


MPS, MRD, and MPP are not displayed as a part of the ladder display.
MPS
MRD
MPP

[Functions]

MPS

(1) Stores in memory the operation result (ON or OFF) immediately prior to the MPS instruction.

(2) Up to 16 MPS instructions can be used successively.
However, only up to 11 can be created in the ladder mode.
If an MPP instruction is used during this process, the number of uses calculated for the MPS instruction will be decremented by one.

MRD

(1) Reads the operation result stored for the MPS instruction, and uses that result to perform the operation in the next step.

MPP

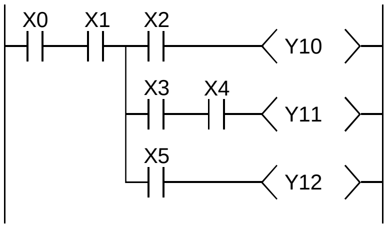
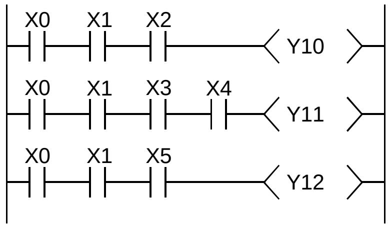
(1) Reads the operation result stored for the MPS instruction, and uses that result to perform the operation in the next step.

(2) Clears the operation results stored by the MPS instruction.

(3) Subtracts 1 from the number of MPS instruction times of use.

POINTS

(1) The following shows ladders both using and not using the MPS, MRD, and MPP instructions.

Ladder Using the MPS, MRD and MPP Instruction.	Ladder not Using MPS, MRD, and MPP Instructions.
	 X0 X1 X2 Y10 X0 X1 X3 X4 Y11 X0 X1 X5 Y12

(2) The MPS and MPP instructions must be used the same number of times. Failure to observe this will not correctly display the ladder in the ladder mode of the peripheral device.

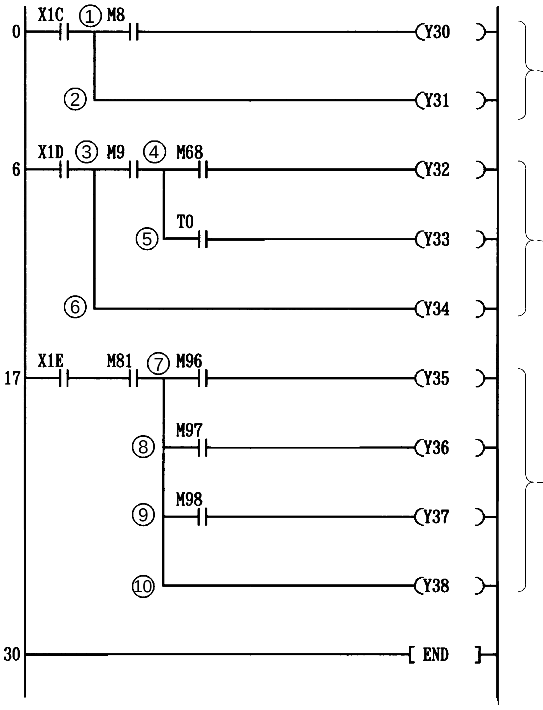
[Operation Errors]

(1) There are no errors associated with the MPS, MRD, or MPP instructions.

[Program Example]

(1) A program using the MPS, MRD, and MPP instructions.

[Ladder Mode]

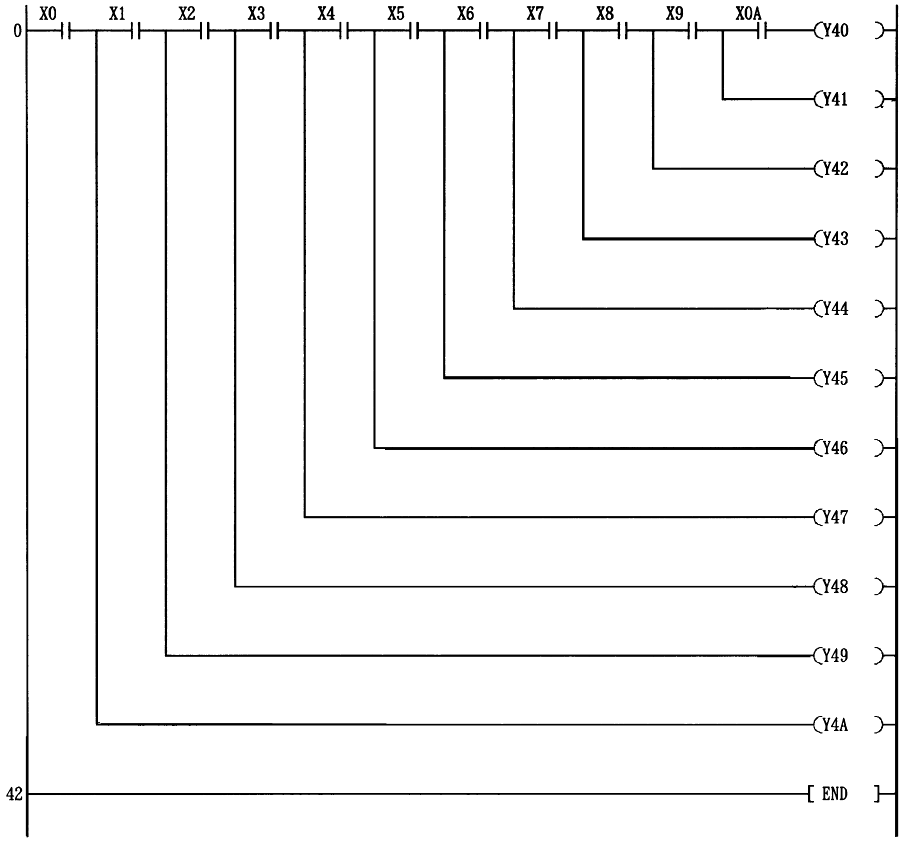


[List Mode]

Steps	Instruction	Device
0	LD	X1C
① 1	MPS	
2	AND	M8
3	OUT	Y30
② 4	MPP	
5	OUT	Y31
6	LD	X1D
③ 7	MPS	
8	AND	M9
④ 9	MPS	
10	AND	M68
11	OUT	Y32
⑤ 12	MPP	
13	AND	T0
14	OUT	Y33
⑥ 15	MPP	
16	OUT	Y34
17	LD	X1E
18	AND	M81
⑦ 19	MPS	
20	AND	M96
21	OUT	Y35
⑧ 22	MRD	
23	AND	M97
24	OUT	Y36
⑨ 25	MRD	
26	AND	M98
27	OUT	Y37
⑩ 28	MPP	
29	OUT	Y38
30	END	

(2) A program using MPS and MPP instructions successively.

[Ladder Mode]



[List Mode]

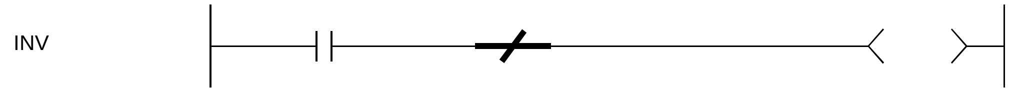
Steps	Instruction	Device
0	LD	X0
1	MPS	
2	AND	X1
3	MPS	
4	AND	X2
5	MPS	
6	AND	X3
7	MPS	
8	AND	X4
9	MPS	
10	AND	X5
11	MPS	
12	AND	X6
13	MPS	
14	AND	X7
15	MPS	
16	AND	X8
17	MPS	
18	AND	X9
19	MPS	
20	AND	X0A
21	OUT	Y40
22	MPP	
23	OUT	Y41
24	MPP	
25	OUT	Y42
26	MPP	
27	OUT	Y43
28	MPP	
29	OUT	Y44
30	MPP	
31	OUT	Y45
32	MPP	
33	OUT	Y46
34	MPP	
35	OUT	Y47
36	MPP	
37	OUT	Y48
38	MPP	
39	OUT	Y49
40	MPP	
41	OUT	Y4A
42	END	

QCPU			QnA	Q4AR
PLC CPU		Process CPU		
Basic	High Performance			
○	○	○	○	○

5.2.3 Operation results inversion (INV)

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant K, H	Other
	Bit	Word		Bit	Word				U
—	—								

[Instruction Symbol] [Execution Condition]



[Functions]

Inverts the operation result immediately prior to the INV instruction.

Operation Result Immediately Prior to the INV Instruction.	Operation Result Following the Execution of the INV Instruction.
OFF	ON
ON	OFF

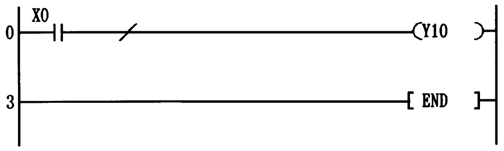
[Operation Errors]

(1) There are no operation errors associated with the INV instruction.

[Program Example]

(1) A program which inverts the X0 ON/OFF data, and outputs from Y10

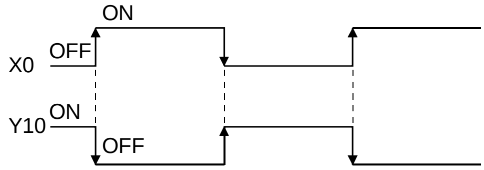
[Ladder Mode]



[List Mode]

Steps	Instruction	Device
0	LD	X0
1	INV	
2	OUT	Y10
3	END	

[Timing Chart]



POINTS

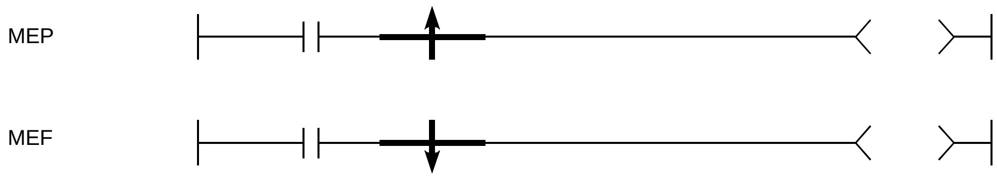
(1) The INV instruction operates based on the results of calculation made until the INV instruction is given. Accordingly, use it in the same position as that of the AND instruction. The INV instruction cannot be used at the LD and OR positions.

QCPU			QnA	Q4AR
PLC CPU		Process CPU		
Basic	High Performance			
○	○	○	○	○

5.2.4 Operation result pulse conversion (MEP, MEF)

Set Data	Usable Devices									
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant K, H	Other
	Bit	Word		Bit	Word				U
—	—								

[Instruction Symbol] [Execution Condition]



[Functions]

MEP

(1) If operation results up to MEP instruction are leading edge (from OFF to ON), goes ON (continuity status).
If operation results up to MEP instruction are anything other than leading edge, goes OFF (non-continuity status).

(2) Use of the MEP instruction simplifies pulse conversion processing when multiple contacts are connected in series.

MEF

(1) If operation results up to MEF instruction are trailing edge (from ON to OFF), goes ON (continuity status).
If operation results up to MEF instruction are anything other than trailing edge, goes OFF (non-continuity status).

(2) Use of the MEF instruction simplifies pulse conversion processing when multiple contacts are connected in series.

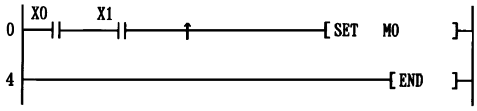
[Operation Errors]

(1) There are no operation errors associated with the MEP or MEF instructions.

[Program Example]

(1) A program which performs pulse conversion on the operation results of X0 and X1:

[Ladder Mode]



[List Mode]

Steps	Instruction	Device
0	LD	X0
1	AND	X1
2	MEP	
3	SET	M0
4	END	

POINTS

(1) The MEP and MEF instructions will occasionally not function properly when pulse conversion is conducted for a contact that has been indexed by a sub-routine program or by the FOR to NEXT instructions.
If pulse conversion is to be conducted for a contact that has been indexed by a sub-routine program or by the FOR to NEXT instructions, use the EGP/EGF instructions.

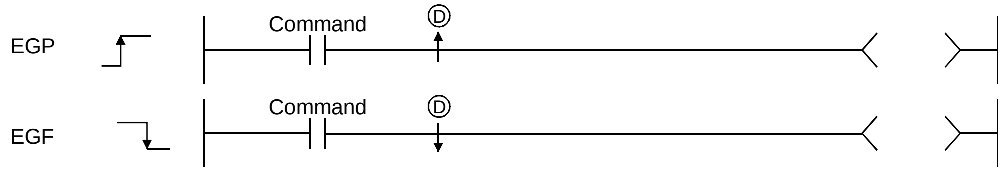
(2) Because the MEP and MEF instructions operate with the operation results immediately prior to the MEP and MEF instructions, the AND instruction should be used at the same position.
The MEP and MEF instructions cannot be used at the LD or OR position.

QCPU					
PLC CPU		Process CPU		QnA	Q4AR
Basic	High Performance				
○	○	○		○	○

5.2.5 Pulse conversion of edge relay operation results (EGP, EGF)

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant K, H	Other
	Bit	Word		Bit	Word				V
Ⓓ	—								○

[Instruction Symbol] [Execution Condition]



[Set Data]

Set Data	Meaning	Data Type
Ⓓ	Edge relay number where operation results are stored	Bit

[Functions]

EGP

(1) Operation results up to the EGP instruction are stored in memory by the edge relay (V).

(2) Goes ON (continuity status) at the leading edge (OFF to ON) of the operation result up to the EGP instruction.
If the operation result up to the EGP instruction is other than a leading edge (i.e., from ON to ON, ON to OFF, or OFF to OFF), it goes OFF (non-continuity status).

(3) The EGP instruction is used for sub-routine programs, and for conducting pulse operations for programs designated by index modification between FOR and NEXT instructions.

(4) The EGP instruction can be used like an AND instruction.

EGF

(1) Operation results up to the EGF instruction are stored in memory by the edge relay (V).

(2) Goes ON at the trailing edge (from ON to OFF) of the operation result up to the EGF instruction.
If the operation result up to the EGF instruction is other than a trailing edge (i.e., from OFF to ON, ON to ON, or OFF to OFF), it goes OFF (non-continuity status).

(3) The EGF instruction is used for sub-routine programs, and for conducting pulse operations for programs designated by index modification between FOR and NEXT instructions.

(4) The EGF instruction can be used like an AND instruction.

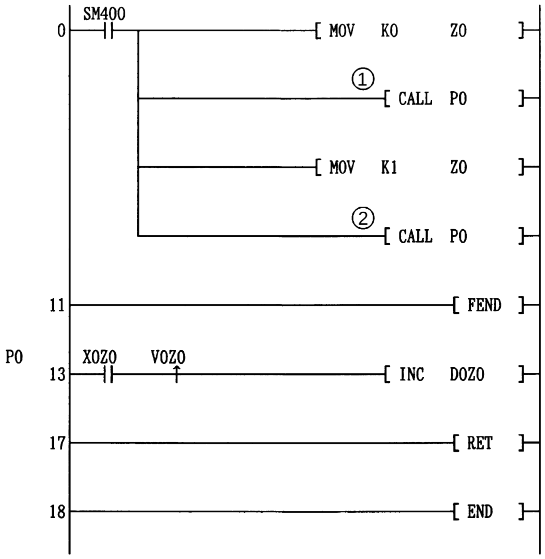
[Operation Errors]

(1) There are no operation errors associated with the EGP or EGF instructions.

[Program Example]

(1) A program containing a subroutine program using an EGP instruction

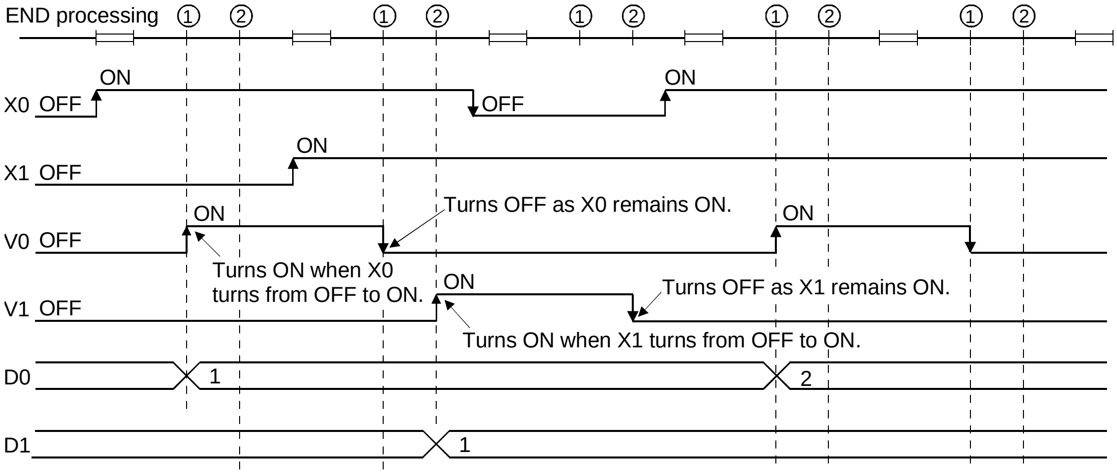
[Ladder Mode]



[List Mode]

Steps	Instruction	Device
0	LD	SM400
1	MOV	K0 Z0
4	CALL	P0
6	MOV	K1 Z0
9	CALL	P0
11	FEND	
12		P0
13	LD	X0Z0
14	EGP	V0Z0
15	INC	D0Z0
17	RET	
18	END	

[Operation]

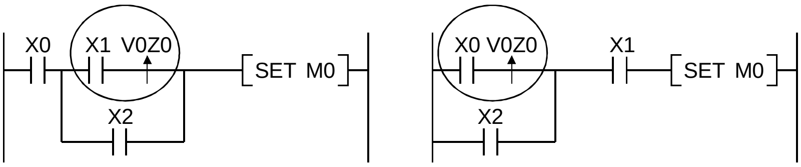

END processing
① ② ① ② ① ② ① ② ① ②
X0 OFF
ON
OFF
ON
X1 OFF
ON
V0 OFF
ON
Turns OFF as X0 remains ON.
ON
Turns ON when X0 turns from OFF to ON.
V1 OFF
ON
Turns OFF as X1 remains ON.
Turns ON when X1 turns from OFF to ON.
D0
1
2
D1
1

POINTS

(1) Since EGP and EGF instructions are executed according to the results of operation performed immediately before the EGP/EGF instruction, these instructions must be used in the same position as the AND instruction (refer to 5.1.1.).

An EGP and EGF instruction cannot be used at the position of an LD or OR instruction.

(2) EGP and EGF instructions cannot be used at the circuit block positions shown below.



QCPU				
PLC CPU		Process CPU	QnA	Q4AR
Basic	High Performance			
○	○	○	○	○

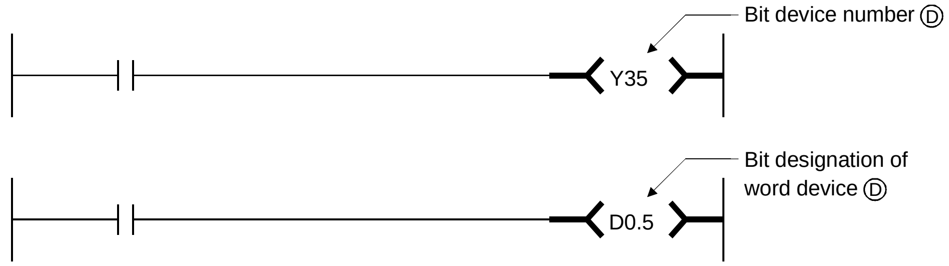
5.3 Out Instructions

5.3.1 Out instructions (excluding timers, counters, and annunciators) (OUT)

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant K, H	Other
	Bit	Word		Bit	Word				DY
Ⓓ	○ (Other than T, C, or F)	○					—		○

[Instruction Symbol] [Execution Condition]

OUT



[Set Data]

Set Data	Meaning	Data Type
Ⓓ	Number of device to be turned ON and OFF	Bit

[Functions]

(1) Operation results up to the OUT instruction are output to the designated device.

Operation Results	When Using Bit Devices			When Bit Designation has been Made for Word Device
	Coil	Contact		Bit Designated
		A Contact	B Contact	
OFF	OFF	Non-continuity	Continuity	0
ON	ON	Continuity	Non-continuity	1

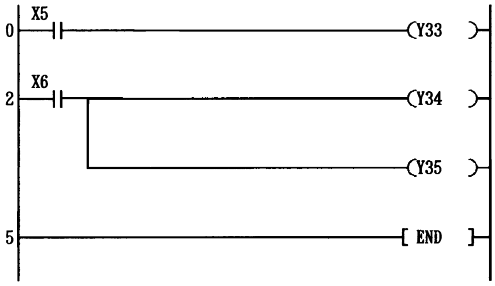
[Operation Errors]

(1) See Section 3.6 for information regarding errors during index modification.

[Program Example]

(1) When bit device is in use

[Ladder Mode]

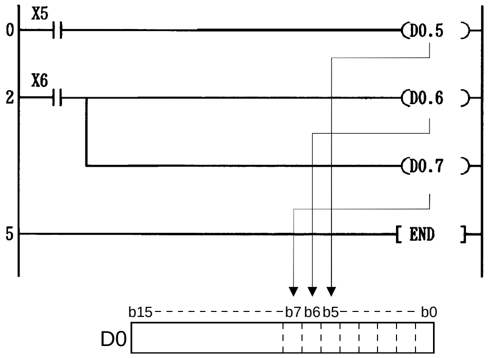


[List Mode]

Steps	Instruction	Device
0	LD	X5
1	OUT	Y33
2	LD	X6
3	OUT	Y34
4	OUT	Y35
5	END	

(2) When bit designation has been made for word device

[Ladder Mode]



[List Mode]

Steps	Instruction	Device
0	LD	X5
1	OUT	D0.5
2	LD	X6
3	OUT	D0.6
4	OUT	D0.7
5	END	

REMARK

The number of basic steps for OUT instructions is as follows:

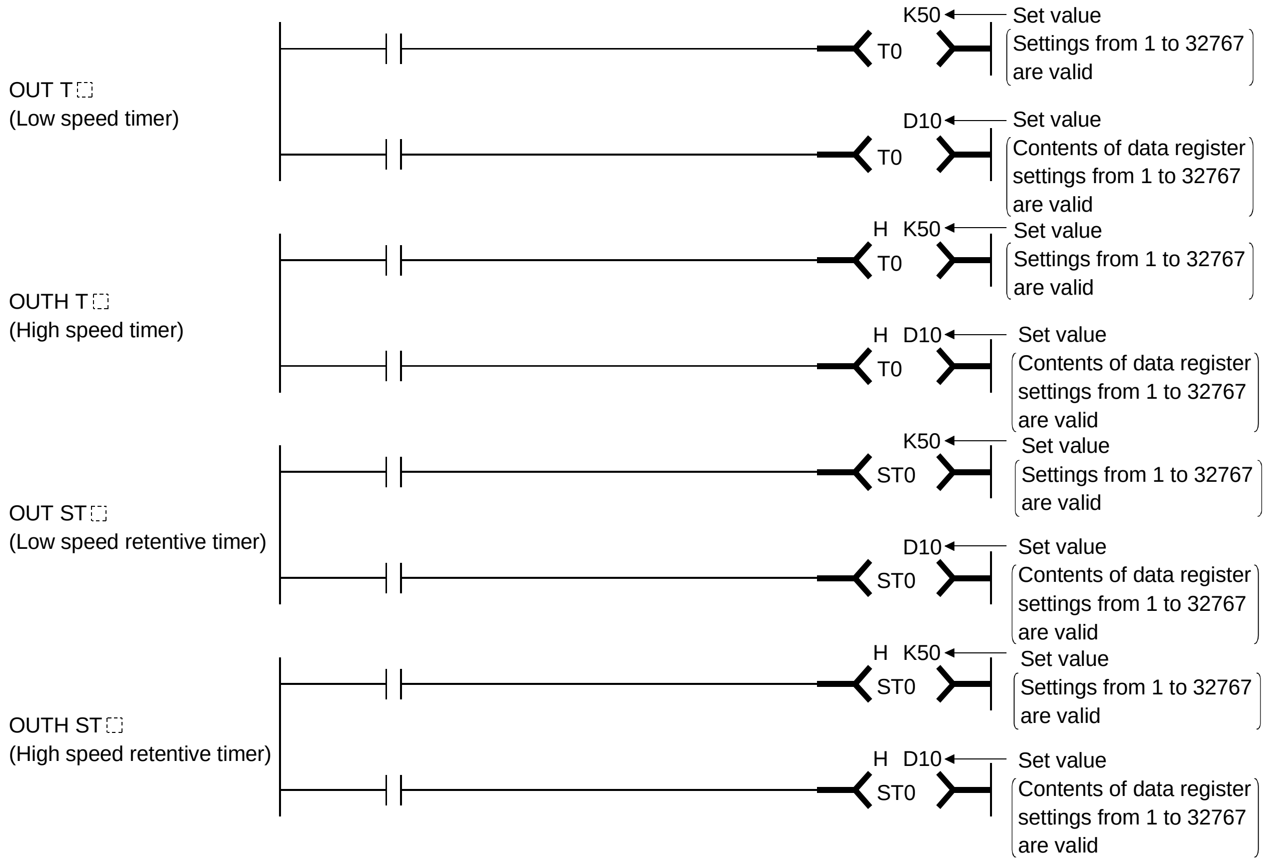
- When using internal device or file register (R) : 1
- When using direct access output (DY) : 2
- When using any other device : 3
  (Including serial number access file register)

QCPU			QnA	Q4AR
PLC CPU		Process CPU		
Basic	High Performance			
○	○	○	○	○

5.3.2 Timers (OUT T, OUTH T)

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant K	Other
	Bit	Word		Bit	Word				
Ⓓ	○ (Only T)	–	–					–	–
Set value	–	○ (Other than T,C)	○ *2	–	○		–	○ *1	–

[Instruction Symbol]   [Execution Condition]


OUT T□
(Low speed timer)
K50
T0
Set value
Settings from 1 to 32767 are valid
D10
T0
Set value
Contents of data register settings from 1 to 32767 are valid
OUTH T□
(High speed timer)
H K50
T0
Set value
Settings from 1 to 32767 are valid
H D10
T0
Set value
Contents of data register settings from 1 to 32767 are valid
OUT ST□
(Low speed retentive timer)
K50
ST0
Set value
Settings from 1 to 32767 are valid
D10
ST0
Set value
Contents of data register settings from 1 to 32767 are valid
OUTH ST□
(High speed retentive timer)
H K50
ST0
Set value
Settings from 1 to 32767 are valid
H D10
ST0
Set value
Contents of data register settings from 1 to 32767 are valid

REMARK

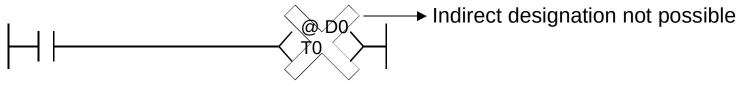
＊1: Timer values can be set only as a decimal constant (K).
Hexadecimal constants (H) and real numbers cannot be used for timer settings.
＊2: The file register cannot be used in the Q00JCPU.

[Set Data]

Set Data	Meaning	Data Type
Ⓓ	Timer number	Bit
Set value	Value set for timer	BIN 16 bits ＊3

POINT

＊3: The value setting for the timer cannot be designated indirectly.



See Section 3.4 for further information on indirect designations.

[Functions]

(1) When the operation results up to the OUT instruction are ON, the timer coil goes ON and the timer counts up to the value that has been set; when the time up status (total numeric value is equal to or greater than the setting value), the contact responds as follows:

A contact	Continuity
B contact	Non-continuity

(2) The contact responds as follows when the operation result up to the OUT instruction is a change from ON to OFF:

Type of Timer	Timer Coil	Present Value of Timer	Prior to Time Up		After Time Up	
			A Contact	B Contact	A Contact	B Contact
Low speed timer High speed timer	OFF	0	Non-continuity	Continuity	Non-continuity	Continuity
Low speed retentive timer High speed retentive timer	OFF	Maintains the present value	Non-continuity	Continuity	Continuity	Non-continuity

＊ The present value is cleared from low speed retentive timers and high speed retentive timers, and the contact is reset, by use of the RST instruction.

(3) To clear the present value of a retentive timer and turn the contact OFF after time up, use the RST instruction.

(4) A negative number (-32768 to -1) cannot be set as the setting value for the timer.
If the setting value is 0, the timer will time out when the time the OUT instruction is executed.

(5) The following processing is conducted when the OUT instruction is executed:
- OUT T□ coil turned ON or OFF
- OUT T□ contact turned ON or OFF
- OUT T□ present value updated

In cases where a JMP instruction or the like is used to jump to an OUT T□ instruction while the OUT T□ instruction is ON, no present value update or contact ON/OFF operation is conducted. Also, if the same OUT T□ instruction is conducted two or more times during the same scan, the present value of the number of repetitions executed will be updated.

(6) Index modification for timer coils or contacts can be conducted only by Z0 or Z1.
Index modification cannot be conducted for the set value for the timer.

REMARKS

(1) The default value for the low speed timer and low speed retentive timer time limit is 100 ms. The time limit for the low speed timer and low speed retentive timer can be set in the parameter mode "PLC system settings" area in increments of 10 ms between the limits of 10 ms to 1 second.

(2) The default value for time limits for the high speed timer and the high speed retentive timer is 10 ms.
The time limits for the high speed timer and the high speed retentive timer can be set in the parameter mode "PLC system settings" area in increments of 1 ms between the limits of 10 ms and 100 ms.

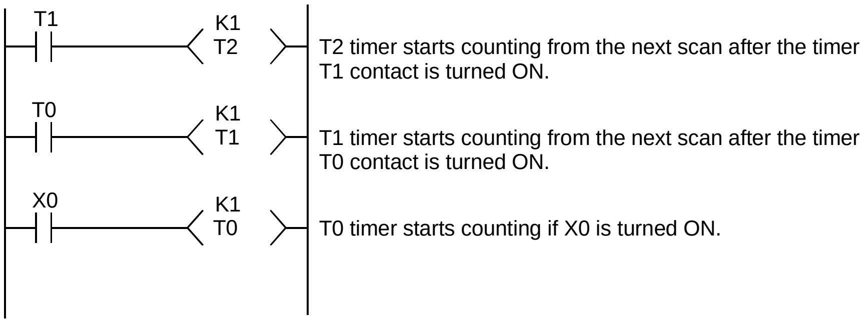
(3) Refer to the User's Manual (Functions Explanation, Programming Fundamentals) of the used CPU module or QnACPU Programming Manual (Fundamentals) for information on timer counting methods.

[Cautions]

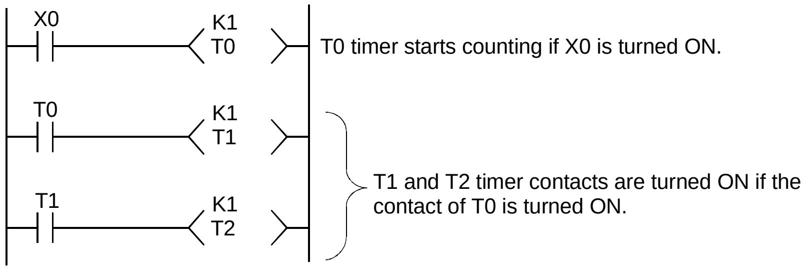
(1) When creating a program in which the operation of the timer contacttriggers the operation of other timer, create the program according to the operation order of the timers - create the program for the timer that operates later first.
In the following cases, all timers go ON at the same scan if the program is created in the order the timers operate,
- If the set value is smaller than a scan time.
- If "1" is set.

Example

- For timers T0 to T2, the program is created in the order the timer operates later.


T1
K1
T2
T2 timer starts counting from the next scan after the timer T1 contact is turned ON.
T0
K1
T1
T1 timer starts counting from the next scan after the timer T0 contact is turned ON.
X0
K1
T0
T0 timer starts counting if X0 is turned ON.

- For timers T0 to T2, the program is created in the order of timer operation.


X0
K1
T0
T0 timer starts counting if X0 is turned ON.
T0
K1
T1
T1
K1
T2
T1 and T2 timer contacts are turned ON if the contact of T0 is turned ON.

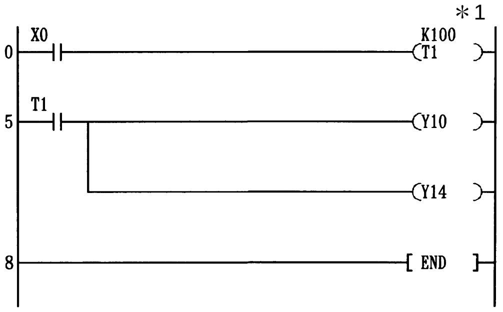
[Operation Errors]

(1) There are no operation errors associated with the OUT T ⌜⌟ instruction.

[Program Example]

(1) The following program turns Y10 and Y14 ON 10 seconds after X0 has gone ON.

[Ladder Mode]

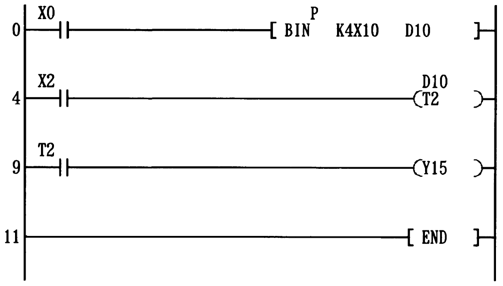


[List Mode]

Steps	Instruction	Device
0	LD	X0
1	OUT	T1 K100
5	LD	T1
6	OUT	Y10
7	OUT	Y14
8	END	

(2) The following program uses the BCD data at X10 to 1F as the timer's set value.

[Ladder Mode]



Converts BCD data at X10 to 1F to BIN and stores at D10

When X2 goes ON, the data stored at D10 is calculated as the set value.

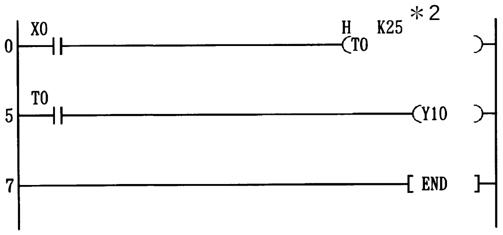
Y15 goes ON when T2 counts out.

[List Mode]

Steps	Instruction	Device
0	LD	X0
1	BINP	K4X10 D10
4	LD	X2
5	OUT	T2 D10
9	LD	T2
10	OUT	Y15
11	END	

(3) The following program turns Y10 ON 250 m after X0 goes ON.

[Ladder Mode]



[List Mode]

Steps	Instruction	Device
0	LD	X0
1	OUTH	T0 K25
5	LD	T0
6	OUT	Y10
7	END	

REMARKS

(1) ＊1: The set value of the low speed timer indicates its default time limit (100 ms).

(2) ＊2: The set value of the high speed timer indicates its default time limit (10 ms)

(3) The number of basic steps of the OUT T ⌜⌟ instruction is 4.

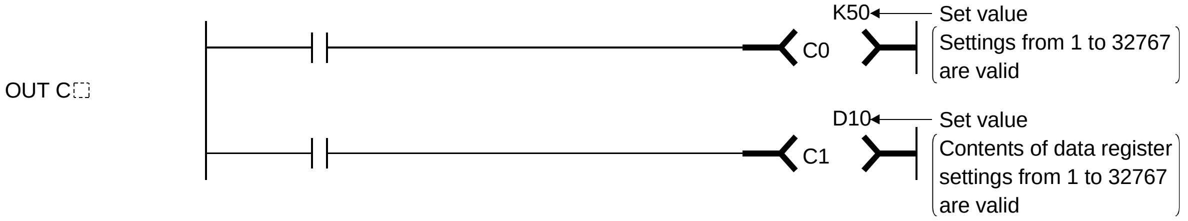
QCPU			QnA	Q4AR
PLC CPU		Process CPU		
Basic	High Performance			
○	○	○	○	○

5.3.3 Counters (OUT C)

Set Data	Usable Devices									
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant K	Other
	Bit	Word		Bit	Word				U
Ⓓ	○ (Only C)	—	—					—	—
Set value	—	○ (Other than T,C)	○ *3	—	○		—	○ *1	—

[Instruction Symbol] [Execution Condition]

OUT C□


K50
C0
Set value
Settings from 1 to 32767 are valid
D10
C1
Set value
Contents of data register settings from 1 to 32767 are valid

[Set Data]

Set Data	Meaning	Data Type
Ⓓ	Counter number	Bit
Set value	Counter set value	BIN 16 bits *2

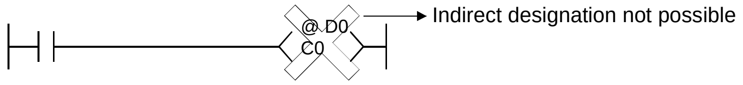
[Functions]

(1) When the operation results up to the OUT instruction change from OFF to ON, 1 is added to the present value (count value) and the count up status (present value = set value), and the contacts respond as follows:

A contact	Continuity
B contact	Non-continuity

POINT

*2: Counter value cannot be set by indirect designation.



See Section 3.4 for further information on indirect designations.

REMARKS

Refer to the User's Manual (Functions Explanation, Programming Fundamentals) of the used CPU module or QnACPU Programming Manual (Fundamentals) for counter counting methods.

*1: Counter value can be set only with a decimal constant (K).
A hexadecimal constant (H) or a real number cannot be used for the counter value setting.

*3: The file register cannot be used in the Q00JCPU.

(2) No count is conducted with the operation results at ON.
(There is no need to perform pulse conversion on count input.)

(3) After the count up status is reached, there is no change in the count value or the contacts until the RST instruction is executed.

(4) A negative number (-32768 to -1) cannot be set as the setting value for the timer.
If the set value is 0, the processing is identical to that which takes place for 1.

(5) Index modification for the counter coil and contact can use only Z0 and Z1.
Index modification cannot be performed for the counter setting.

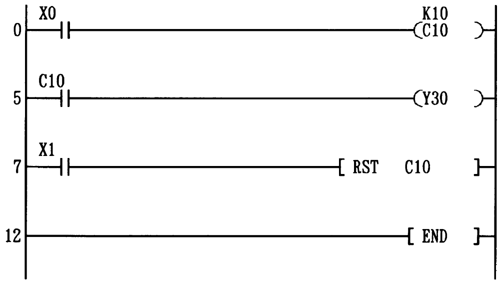
[Operation Errors]

(1) There are no operation errors associated with the OUT C □ instruction.

[Program Example]

(1) The following program turns Y30 ON after X0 has gone ON 10 times, and resets the counter when X1 goes ON.

[Ladder Mode]

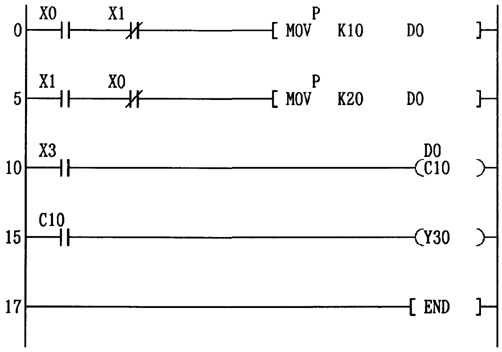


[List Mode]

```
Steps   Instruction   Device
  0     LD            X0
  1     OUT           C10
                      K10
  5     LD            C10
  6     OUT           Y30
  7     LD            X1
  8     RST           C10
 12     END
```

(2) The following program sets the value for C10 at 10 when X0 goes ON, and at 20 when X1 goes ON.

[Ladder Mode]



Stores 10 at D0 when X0 goes ON

Stores 20 at D0 when X1 goes ON

C10 takes data stored at D0 as set value, and counts

Y30 goes ON when C10 reaches count out state

[List Mode]

```
Steps   Instruction   Device
  0     LD            X0
  1     ANI           X1
  2     MOVP          K10
                      D0
  5     LD            X1
  6     ANI           X0
  7     MOVP          K20
                      D0
 10     LD            X3
 11     OUT           C10
                      D0
 15     LD            C10
 16     OUT           Y30
 17     END
```

REMARK

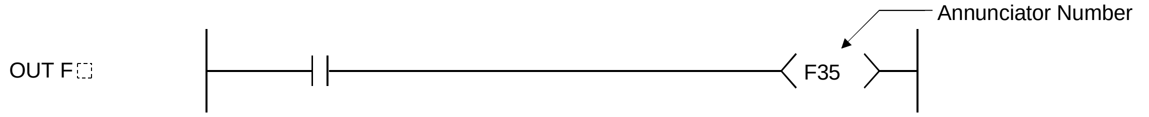
The number of basic steps of the OUT C □ instruction is 4.

QCPU			QnA	Q4AR
PLC CPU		Process CPU		
Basic	High Performance			
○	○	○	○	○

5.3.4 Annunciator output (OUT F)

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant K, H	Other
	Bit	Word		Bit	Word				
Ⓓ	○ (Only F)	—							

[Instruction Symbol] [Execution Condition]



[Set Data]

Set Data	Meaning	Data Type
Ⓓ	Number of annunciator to be turned ON	Bit

[Functions]

(1) Operation results up to the OUT instruction are output to the designated annunciator.

(2) The following responses occur when an annunciator (F) is turned ON.

[With Q3A, Q4A, or Q4ARCPU]

- The annunciator number is displayed at the LED display device on the front of the CPU module, and the "USER" LED goes ON.
- The annunciator numbers which are ON (F numbers) are stored in special registers (SD64 to SD79).
- The value of SD63 is incremented by 1.

[With CPUs other than above]

- The "USER" LED goes ON.
- The annunciator numbers which are ON (F numbers) are stored in special registers (SD64 to SD79).
- The value of SD63 is incremented by 1.

(3) If the value of SD63 is 16 (which happens when 16 annunciators are already ON), even if a new annunciator is turned ON, its number will not be stored at SD64 to SD79.

(4) The following responses occur when the annunciator is turned OFF by the OUT instruction.

[With Q3A, Q4A, or Q4ARCPU]

- The coil goes OFF, but there are no changes in the LED display device on the front of the CPU module, the status of the "USER" LED, and the contents of the values stored in SD63 to SD79.
- Use the RST F□ instruction to turn OFF the LED display device on the front of the CPU module and "USER" LED, and to delete the annunciator which was turned OFF by the OUT F□ instruction from SD63 to SD79.

[With CPUs other than above]

- The coil goes OFF, but there are no changes in the status of the "USER" / "ERR." LED and the contents of the values stored in SD63 to SD79.
- Use the RST F□ instruction to turn OFF the "USER" / "ERR." LED and to delete the annunciator which was turned OFF by the OUT F□ instruction from SD63 to SD79.

[Operation Errors]

(1) There are no operation errors associated with the OUT F□ instruction.

REMARKS

(1) Refer to User's Manual (Functions Explanation, Programming Fundamentals) of the used CPU module or QnACPU Programming Manual (Fundamentals) for details of annunciators

(2) The number of basic steps for the OUT module F□ instruction is 4.

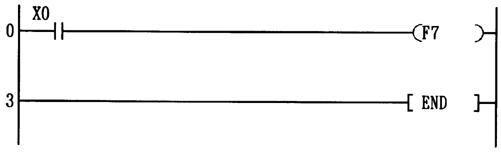
(3) The table below shows which CPU module features either the LED display device on front of the CPU module or "USER" LED

Type of LED	CPU module Type Name
LED display device	Q3A, Q4A, Q4AR
"USER" LED	Q2A(S1), Q2AS(S1), Q2ASH(S1), High Performance model QCPU
"ERR." LED	Basic model QCPU

[Program Example]

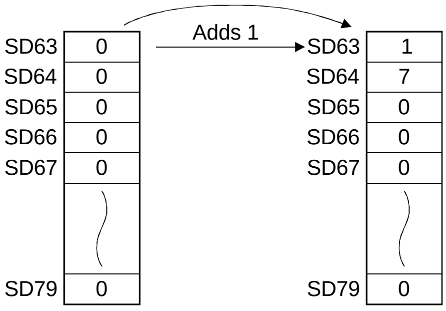
(1) The following program turns F7 ON when X0 goes ON, and stores the value 7 from SD64 to SD79.

[Ladder Mode]



[List Mode]

Steps	Instruction	Device
0	LD	X0
1	OUT	F7
3	END	

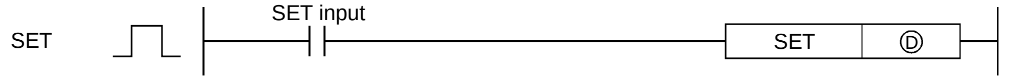


QCPU			QnA	Q4AR
PLC CPU		Process CPU		
Basic	High Performance			
○	○	○	○	○

5.3.5 Setting devices (except for annunciators) (SET)

Set Data	Usable Devices									
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J[ ]\[ ]		Special Function Module U[ ]\G[ ]	Index Register Zn	Constant K, H	Other	
	Bit	Word		Bit	Word				BL	DY
Ⓓ	○	○		○	○		–	–	○	○

[Instruction Symbol] [Execution Condition]



[Set Data]

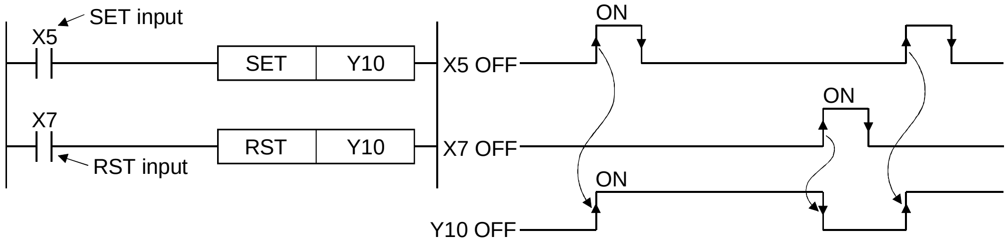
Set Data	Meaning	Data Type
Ⓓ	Bit device number to be set (ON)/Word device bit designation	Bit

[Functions]

(1) When SET input is ON, the designated devices respond as follows:

Device	Device Status
Bit device	Coils and contacts turned ON
When bit designation has been made for word device	Designation bit set at 1

(2) Devices turned ON will stay ON even if SET input goes to OFF.
Devices turned ON by the SET instruction can be turned OFF by the RST instruction.



(3) Device status does not change when SET input is OFF.

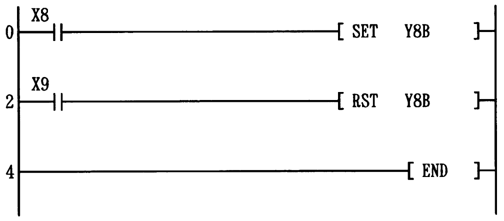
[Operation Errors]

(1) There are no operation errors associated with the SET instruction.

[Program Example]

(1) The following program sets Y8B (ON) when X8 goes ON, and resets Y8B (OFF) when X9 goes ON.

[Ladder Mode]

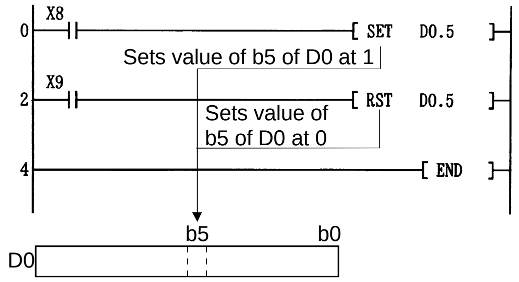


[List Mode]

Steps	Instruction	Device
0	LD	X8
1	SET	Y8B
2	LD	X9
3	RST	Y8B
4	END	

(2) The following program sets the value of D0 bit 5 (b5) to 1 when X8 goes ON, and set the bit value to 0 when X9 goes ON.

[Ladder Mode]



[List Mode]

Steps	Instruction	Device
0	LD	X8
1	SET	D0.5
2	LD	X9
3	RST	D0.5
4	END	

REMARK

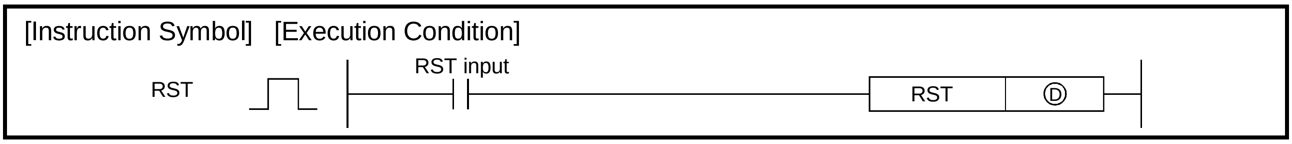
The basic SET instructions are as follows:

• When internal device or file register (R0 to R32767) are in use	: Step 1
• When direct access output (DY) or SFC program device (BL) are in use	: Step 2
• When timer (T) or counter (C) are in use	: Step 4
• When some other device is in use	: Step 3

QCPU			QnA	Q4AR
PLC CPU		Process CPU		
Basic	High Performance			
○	○	○	○	○

5.3.6 Resetting devices (except for annunciators) (RST)

Set Data	Usable Devices									
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J⁅ ⁆\⁅ ⁆		Special Function Module U⁅ ⁆\G⁅ ⁆	Index Register Zn	Constant K, H	Other	
	Bit	Word		Bit	Word				BL	DY
Ⓓ	○	○		○	○		○	–	–	○



[Set Data]

Set Data	Meaning	Data Type
Ⓓ	Bit device number to be reset/ Word device bit designation	Bit
	Word device number to be reset	BIN 16 bits

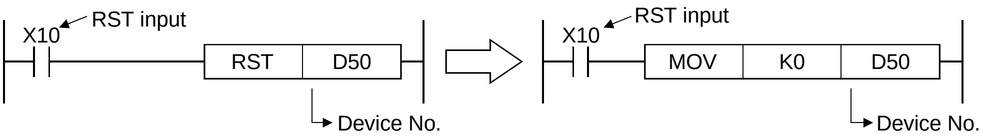
[Functions]

(1) Designated devices respond as follows when RST input is turned ON:

Device	Device Status
Bit device	Turns coils and contacts OFF
Timers and counters	Sets the present value to 0, and turns coils and contacts OFF
When bit designation has been made for word device	Sets value of designated bit to 0
Word devices other than timers and counters	Sets contact to 0

(2) Device status does not change when RST input goes OFF.

(3) The functions of the word devices designated by the RST instruction are identical to the following ladder:



[Operation Errors]

(1) There are no operation errors associated with the RST instruction.

REMARK

1) The basic number of steps of the RST instruction is as follows.

a) For bit processing

• Internal device (bit to be specified by bit device or word device)	: 1
• Direct output	: 2
• Timer, counter	: 4
• Other than above	: 3

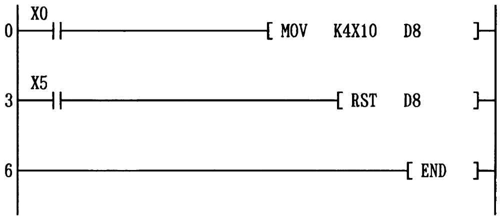
b) For word processing

• Internal device (word to be specified by bit device)	: 2
• Index resister	: 2
• Other than above	: 3

[Program Example]

(1) The following program sets the value of the data register to 0.

[Ladder Mode]



When X0 goes ON, the contents of X10 to 1F are stored at D8

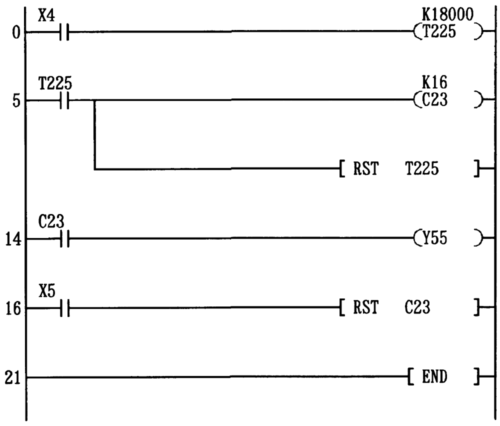
When X5 goes ON, the value of D8 is set to 0

[List Mode]

```
Steps    Instruction    Device
  0      LD             X0
  1      MOV            K4X10
                        D8
  3      LD             X5
  4      RST            D8
  6      END
```

(2) The following program resets the 100 ms retentive timer and counter.

[Ladder Mode]



When T225 is set at the retentive timer, the ON time for X4 is 30 minutes, then T225 goes ON.

Counts the number of times T225 goes ON.

When the T225 contact goes ON, the T225 coil, contact, and present value are reset.

When C23 reaches the count out state, Y55 goes ON.

Resets C23 when X5 goes ON.

[List Mode]

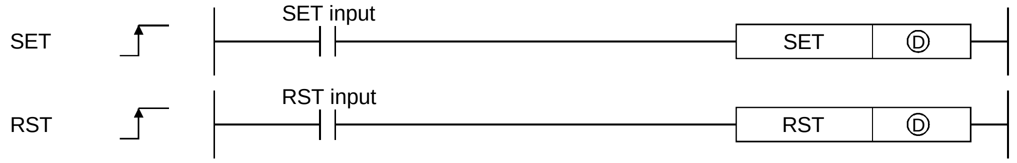
```
Steps    Instruction    Device
  0      LD             X4
  1      OUT            T225
                        K18000
  5      LD             T225
  6      OUT            C23
                        K16
 10      RST            T225
 14      LD             C23
 15      OUT            Y55
 16      LD             X5
 17      RST            C23
 21      END
```

QCPU			QnA	Q4AR
PLC CPU		Process CPU		
Basic	High Performance			
○	○	○	○	○

5.3.7 Setting and resetting the annunciators (SET F, RST F)

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant K, H	Other
	Bit	Word		Bit	Word				U
Ⓓ	○ (Only F)	–							

[Instruction Symbol] [Execution Condition]



[Set Data]

Instruction Name	Set Data	Meaning	Data Type
SET	Ⓓ	Number of annunciator to be set (F number)	Bit
RST	Ⓓ	Number of annunciator to be reset (F number)	

[Functions]

SET

(1) Annunciator designated by Ⓓ goes ON when SET input goes ON.

(2) The following responses occur when the annunciator (F) goes ON:
- The annunciator number is displayed at the LED display device at the front of the CPU module, or the "USER" LED goes ON.
- The numbers (F numbers) of the annunciators turned ON are stored successively at the special registers (SD63 to SD79).
- The value of SD63 is incremented by 1.

(3) If the value of SD63 is 16 (which happens when 16 annunciators are already ON), even if a new annunciator is turned ON, its number will not be stored at SD64 to SD79.

RST

(1) Annunciators designated by Ⓓ are turned OFF when RST input goes ON.

(2) The annunciator numbers (F numbers) of annunciators that have gone OFF are deleted from the special registers (SD64 to SD79), and the value of SD63 is decremented by 1.

REMARK

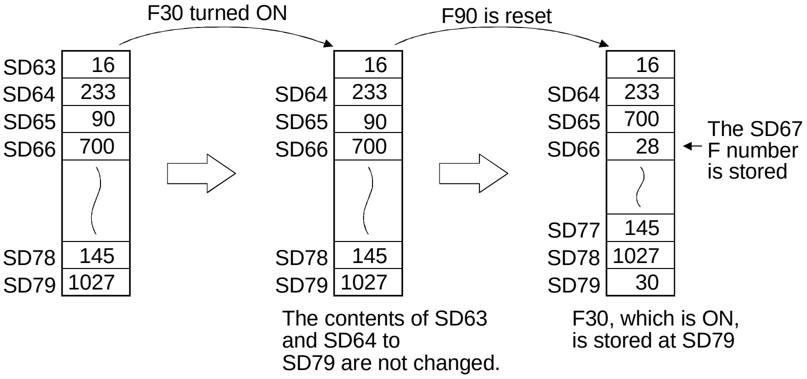
(1) Refer to the User's Manual (Functions Explanation, Programming Fundamentals) of the used CPU module or QnACPU Programming Manual (Fundamentals), Section 4.2.5, for details of annunciators.

(2) The table below shows which CPU module features either the LED display device on front of the CPU module or "USER" LED

Type of LED	CPU module Type Name
LED display device	Q3A, Q4A, Q4AR
"USER LED"	Q2A(S1), Q2AS(S1), Q2ASH(S1), QCPU

(3) If, when the value of SD63 is 16, and annunciator numbers are deleted from SD64 to SD79 by use of the RST instruction, annunciators whose numbers are not registered in SD64 to SD79 are then turned ON, the numbers of these annunciators will be registered. If all annunciator numbers from SD64 to SD79 are turned OFF, the LED display device on the front of the CPU module, or the "USER" LED, will be turned OFF.

[Operations which take place when SD63 is 16]



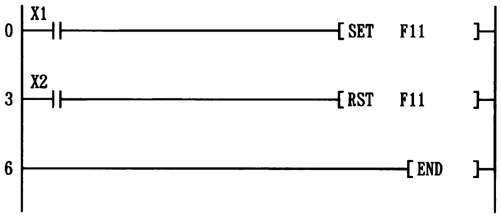
[Operation Errors]

(1) There are no operation errors associated with the SET F□ or RST F□ instructions.

[Program Example]

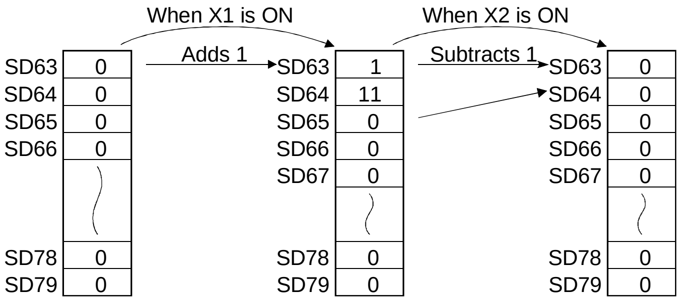
(1) The following program turns annunciator F11 ON when X1 goes ON, and stores the value 11 at the special register (SD64 to SD79).Further, the program resets annunciator F11 if X2 goes ON, and deletes the value 11 from the special registers (SD64 to SD79).

[Ladder Mode]



[List Mode]

Steps	Instruction	Device
0	LD	X1
1	SET	F11
3	LD	X2
4	RST	F11
6	END	

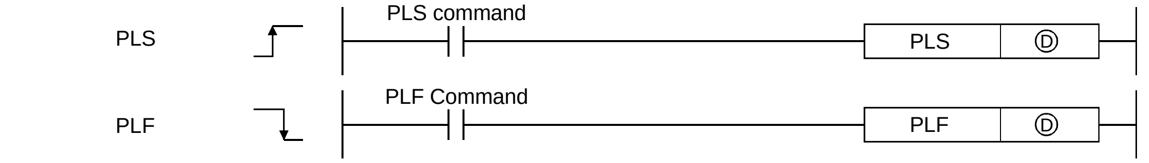


QCPU			QnA	Q4AR
PLC CPU		Process CPU		
Basic	High Performance			
○	○	○	○	○

5.3.8 Leading edge and trailing edge output (PLS, PLF)

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant K, H	Other
	Bit	Word		Bit	Word				DY
Ⓓ	○						—		○

[Instruction Symbol] [Execution Condition]



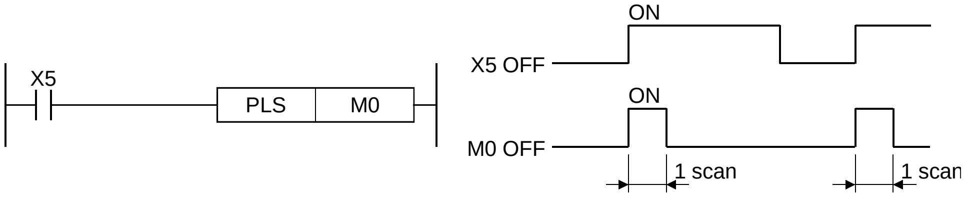
[Set Data]

Set Data	Meaning	Data Type
Ⓓ	Pulse conversion device	Bit

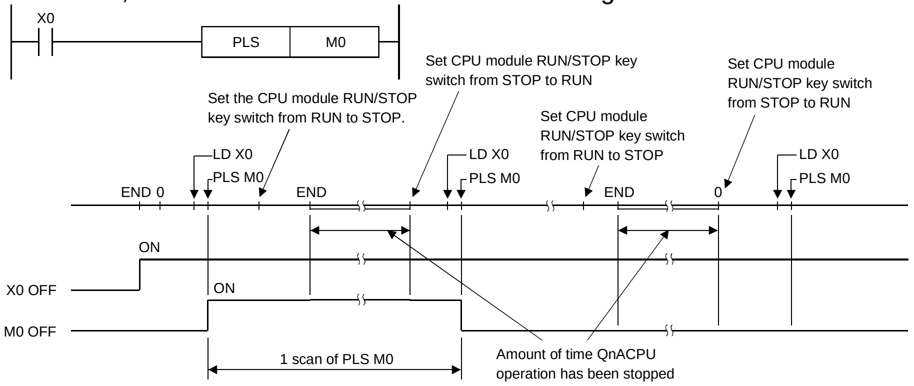
[Functions]

PLS

(1) When turned from OFF to ON, the PLS command turns ON the specified device, and other than when turned from OFF to ON (i.e. from ON to ON, from ON to OFF or from OFF to OFF), it turns OFF the specified device.
When there is one PLS instruction for the device designated by Ⓓ during one scan, the specified device turns ON one scan.
See Section 3.9 for the operation to be performed when the PLS instruction for the same device is executed more than once during one scan.



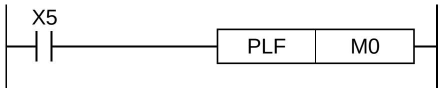
(2) If the RUN/STOP key switch is changed from RUN to STOP after the execution of the PLS instruction, the PLS instruction will not be executed again even if the switch is set back to RUN.

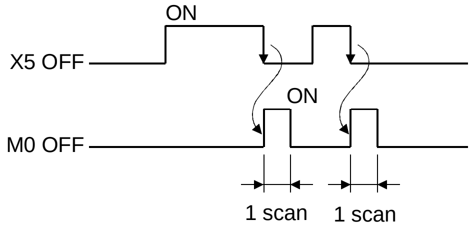

X0
PLS
M0
Set CPU module RUN/STOP key switch from STOP to RUN
Set CPU module RUN/STOP key switch from STOP to RUN
Set the CPU module RUN/STOP key switch from RUN to STOP.
Set CPU module RUN/STOP key switch from RUN to STOP
LD X0
PLS M0
LD X0
PLS M0
LD X0
PLS M0
END 0
END
END
0
ON
X0 OFF
ON
M0 OFF
1 scan of PLS M0
Amount of time QnACPU operation has been stopped

(3) When a latch relay (L) is specified for the PLS command, switching power OFF with the latch relay (L) ON and then switching it ON again executes the PLS instruction to turn ON the specified device since the PLS command turns from OFF to ON at the first scan.
The device turned ON at the first scan after power-ON turns OFF at the next PLS instruction.

PLF

(1) When turned from ON to OFF, the PLF command turns ON the specified device, and other than when turned from ON to OFF (i.e. from OFF to OFF, from OFF to ON or from ON to ON), it turns OFF the specified device.
When there is one PLF instruction for the device designated by Ⓓ during one scan, the specified device turns ON one scan.
See Section 3.9 for the operation to be performed when the PLF instruction for the same device is executed more than once during one scan.





(2) If the RUN/STOP key switch is changed from RUN to STOP after the execution of the PLF instruction, the PLF instruction will not be executed again even if the switch is set back to RUN.

POINT

Note that the device designated by Ⓓ may be ON more than one scan if the PLS or PLF instruction is jumped by the CJ instruction or if the subroutine program where the PLS/PLF instructon was executed was not called by the CALL instruction.

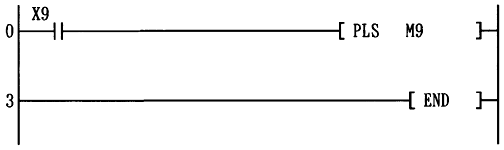
[Operation Errors]

(1) There are no operation errors associated with the PLS or PLF instructions.

[Program Example]

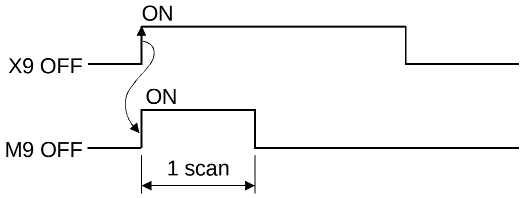
(1) The following program executes the PLS instruction when X9 goes ON.

[Ladder Mode]



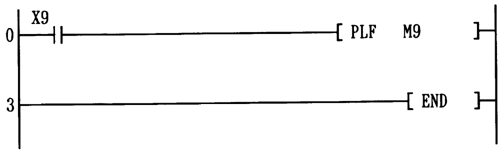
[List Mode]

Steps	Instruction	Device
0	LD	X9
1	PLS	M9
3	END	



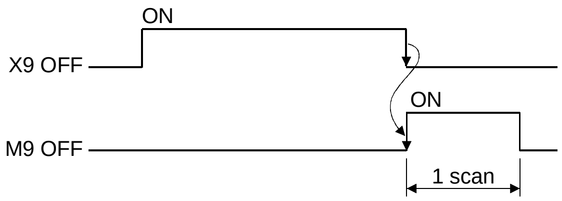
(2) The following program executes the PLF instruction when X9 goes OFF.

[Ladder Mode]



[List Mode]

Steps	Instruction	Device
0	LD	X9
1	PLF	M9
3	END	

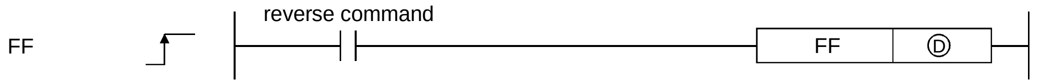


QCPU				QnA	Q4AR
PLC CPU		Process CPU			
Basic	High Performance				
○	○	○		○	○

5.3.9 Bit device output reverse (FF)

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant K, H	Other
	Bit	Word		Bit	Word				DY
Ⓓ	○						—		○

[Instruction Symbol] [Execution Condition]



[Set Data]

Set Data	Meaning	Data Type
Ⓓ	Device number to reverse	Bit

[Functions]

(1) The status of the device designated by Ⓓ is reversed when the inversion command goes from OFF to ON.

Device	Device Status	
	Prior to FF execution	After FF execution
Bit device	OFF	ON
	ON	OFF
Designation of word device	0	1
	1	0

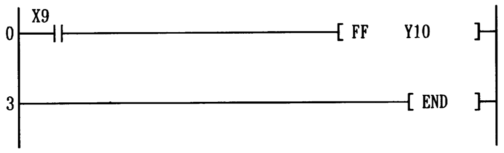
[Operation Errors]

(1) There are no operation errors associated with the FF instruction.

[Program Example]

(1) The following program reverses the output of Y10 when X9 goes ON.

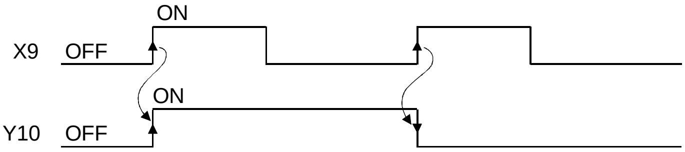
[Ladder Mode]



[List Mode]

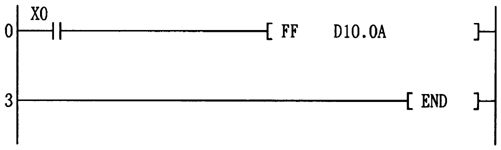
Steps	Instruction	Device
0	LD	X9
1	FF	Y10
3	END	

[Timing Chart]



(2) The following program reverses b10 (bit 10) of D10 when X0 goes ON.

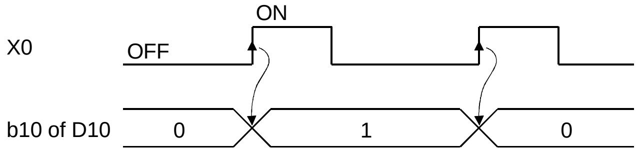
[Ladder Mode]



[List Mode]

Steps	Instruction	Device
0	LD	X0
1	FF	D10.0A
3	END	

[Timing Chart]

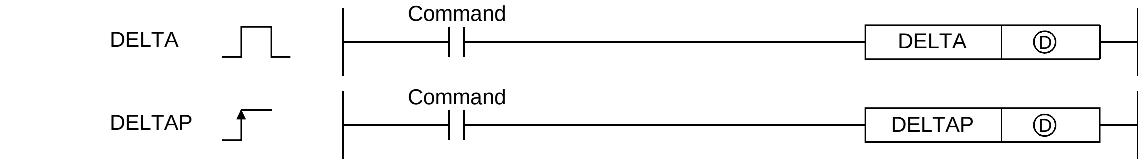


QCPU			QnA	Q4AR
PLC CPU		Process CPU		
Basic	High Performance			
○	○	○	○	○

5.3.10 Pulse conversion of direct output (DELTA, DELTAP)

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J⃞\⃞		Special Function Module U⃞\G⃞	Index Register Zn	Constant K, H	Other
	Bit	Word		Bit	Word				DY
Ⓓ	–								○

[Instruction Symbol] [Execution Condition]



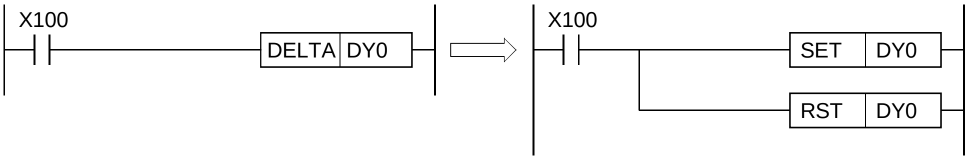
[Set Data]

Set Data	Meaning	Data Type
Ⓓ	Bit for which pulse conversion is to be conducted	Bit

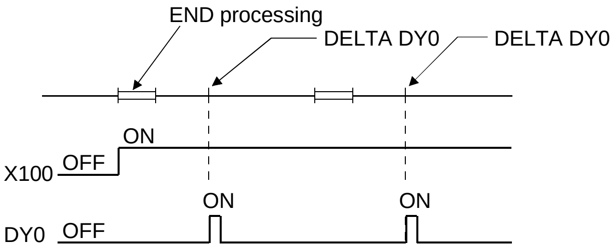
[Functions]

(1) Conducts pulse output of direct access output (DY) designated by Ⓓ.
If DELTA DY0 has been designated, the resulting operation will be identical to the ladder shown below, which uses the SET/RST instructions.

[Ladder created by the DELTA instructions] [Ladder using the SET/RST instruction]



[Operation]



(2) The DELTA (P) instruction is used by commands for leading edge execution for an intelligent function module/special function module.

[Operation Errors]

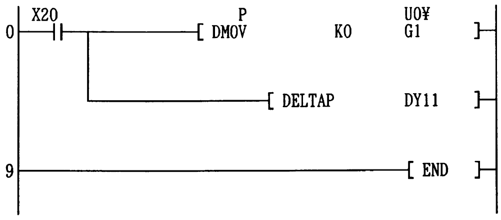
(1) In the following cases an operation error occurs, the error flag (SM0) turns ON, and an error code is stored at SD0.

- A direct access output number designated by Ⓓ has exceeded the CPU output range. (Error code: 4101)

[Program Example]

(1) The following program presets CH1 of the AD61 mounted at slot 0 of the main base unit, when X20 goes ON.

[Ladder Mode]



Stores preset value (0) at addresses 1 and 2 of the AD61 buffer memory

Output of preset instruction

[List Mode]

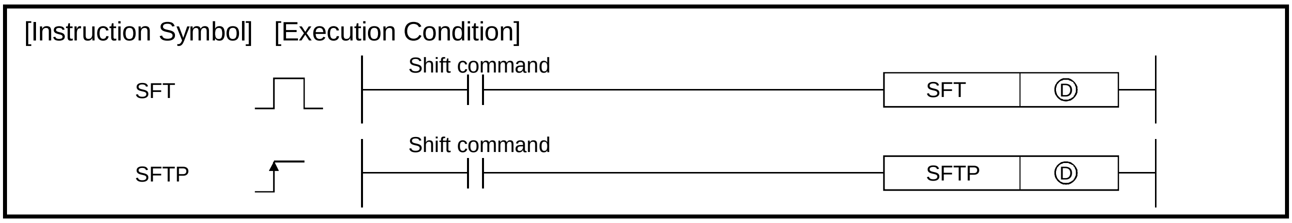
Steps	Instruction	Device
0	LD	X20
1	DMOVP	K0 U0¥G1
6	DELTAP	DY11
9	END	

QCPU			QnA	Q4AR
PLC CPU		Process CPU		
Basic	High Performance			
○	○	○	○	○

5.4 Shift Instruction

5.4.1 Bit device shift (SFT, SFTP)

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant K, H	Other
	Bit	Word		Bit	Word				DY
Ⓓ	○ (Other than TC)						—		○



[Set Data]

Set Data	Meaning	Data Type
Ⓓ	Number of device to shift	Bit

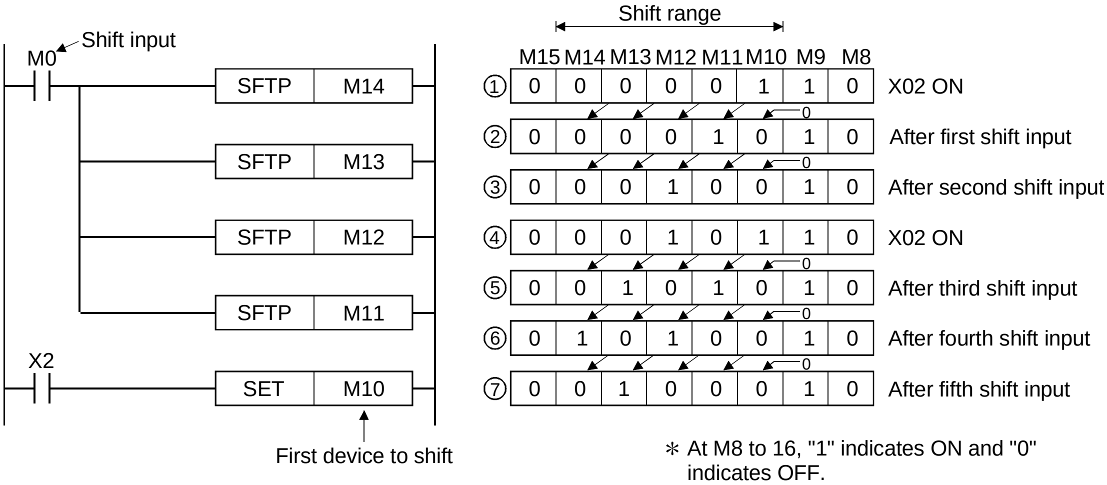
[Functions]

(1) When bit device is used

(a) Shifts to a device designated by Ⓓ the ON/OFF status of the device immediately prior to the one designated, and turns the prior device OFF.
For example, if M11 has been designated by the SFT instruction, when the SFT instruction is executed, it will shift the ON/OFF status of M10 to M11, and turn M10 OFF.

(b) Turn the first device to be shifted ON with the SET instruction.

(c) When the SFT and SFTP are to be used consecutively, the program starts from the device with the larger number.

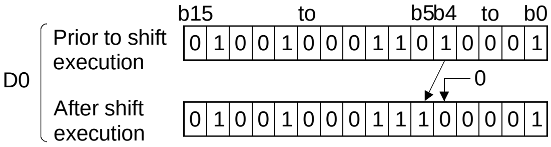


	M15	M14	M13	M12	M11	M10	M9	M8	
①	0	0	0	0	0	1	1	0	X02 ON
②	0	0	0	0	1	0	1	0	After first shift input
③	0	0	0	1	0	0	1	0	After second shift input
④	0	0	0	1	0	1	1	0	X02 ON
⑤	0	0	1	0	1	0	1	0	After third shift input
⑥	0	1	0	1	0	0	1	0	After fourth shift input
⑦	0	0	1	0	0	0	1	0	After fifth shift input

∗ At M8 to 16, "1" indicates ON and "0" indicates OFF.

(2) When word device bit designation is used

(a) Shifts to a bit in the device designated by Ⓓ the 1/0 status of the bit immediately prior to the one designated, and turns the prior bit to 0.
For example, if D0.5 (bit 5 [b5] of D0) has been designated by the SFT instruction, when the SFT instruction is executed, it will shift the 1/0 status of b4 of D0 to b5, and turn b4 to 0.



D0	b15							to				b5	b4		to	b0
Prior to shift execution	0	1	0	0	1	0	0	0	1	1	0	1	0	0	0	1
After shift execution	0	1	0	0	1	0	0	0	1	1	1	0	0	0	0	1

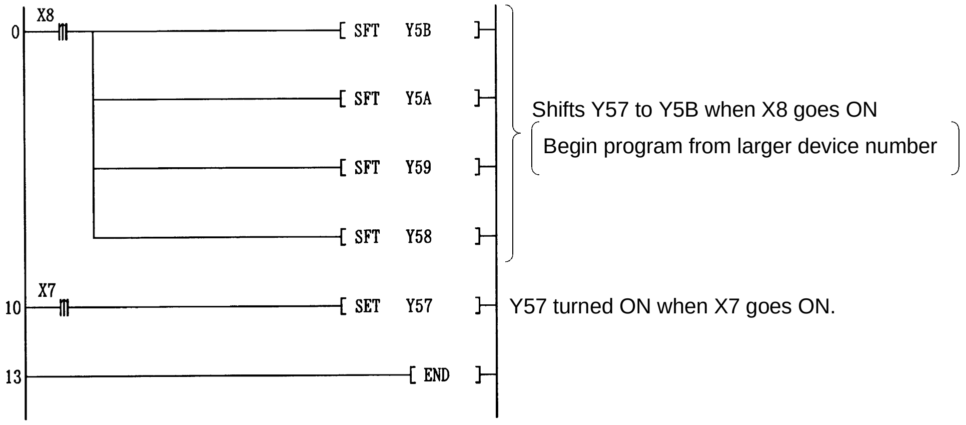
[Operation Errors]

(1) There are no operation errors associated with the SFT (P) instruction.

[Program Example]

(1) The following program shifts Y57 to Y5B when X8 goes ON.

[Ladder Mode]

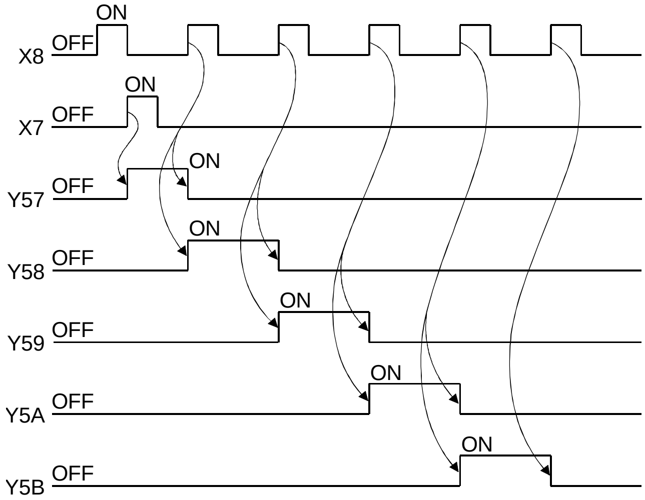


Shifts Y57 to Y5B when X8 goes ON

Begin program from larger device number

Y57 turned ON when X7 goes ON.

[Timing Chart]



[List Mode]

Steps	Instruction	Device
0	LDP	X8
2	SFT	Y5B
4	SFT	Y5A
6	SFT	Y59
8	SFT	Y58
10	LDP	X7
12	SET	Y57
13	END	

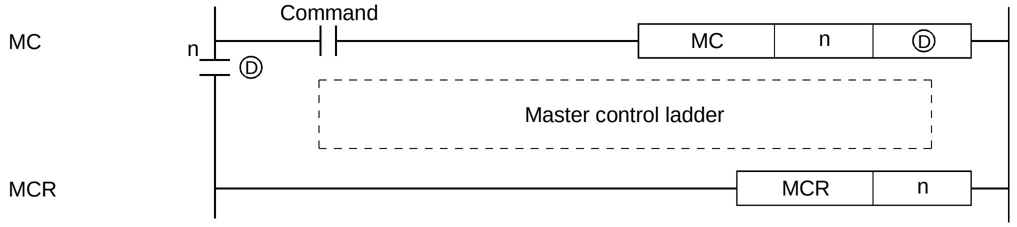
QCPU			QnA	Q4AR
PLC CPU		Process CPU		
Basic	High Performance			
○	○	○	○	○

5.5 Master Control Instructions

5.5.1 Setting and resetting the master control (MC, MCR)

Set Data	Usable Devices									
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant K, H	Other	
	Bit	Word		Bit	Word				N	DY
n	—						—		○	—
Ⓓ	○						—		—	○

[Instruction Symbol] [Execution Condition]



[Set Data]

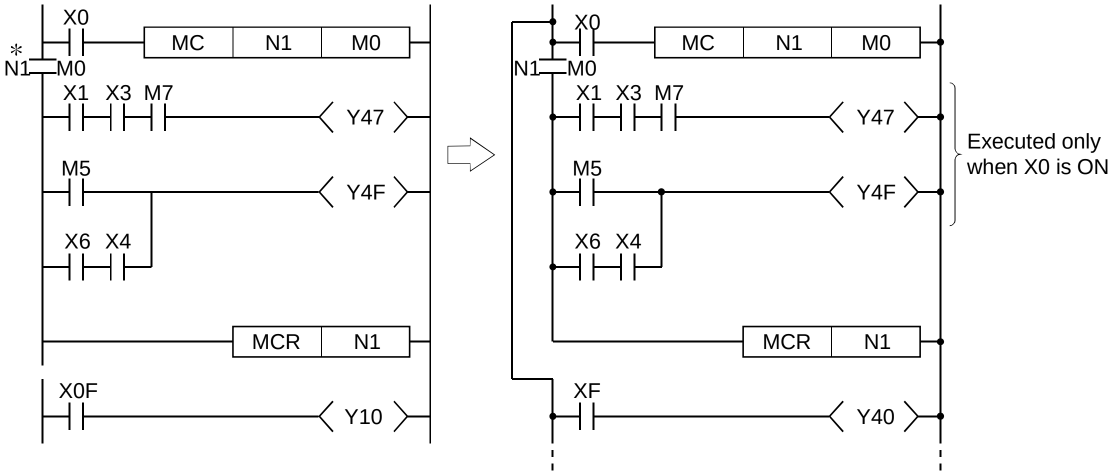
Set Data	Meaning	Data Type
n	Nesting (N0 to N14)	Nesting
Ⓓ	Number of device to turn ON	Bit

[Functions]

The master control instruction is used to enable the creation of highly efficient ladder switching sequence programs, through the opening and closing of a common bus for ladders.

A ladder using the master control would look as shown below:

[Ladder as displayed in GPP Ladder Mode] [Ladder as it actually operates]



REMARK

＊: When programming in the ladder mode of a peripheral device, it is not necessary to input contacts on the vertical bus.
These will be automatically displayed when the "conversion" operation is conducted after the creation of the ladder and then “read” mode is set.

MC

(1) If the ON/OFF command of the MC instruction is ON when master control is commenced, the operation result between the MC instruction and MCR instruction will be exactly as the instruction (ladder) shows.
If the MC ON/OFF indicator is OFF, the operation result between the MC and MCR instructions will be as shown below:

Device		Device Status
High speed timer Low speed timer		Count value goes to 0, coils and contacts all go OFF
High speed retentive timer Low speed retentive timer Counter		Coils go OFF, but counter values and contacts all maintain current status.
Devices in OUT instruction		All turned OFF
SET, RST SFT Basic, Application	Devices in the following instructions:	Maintain current status

(2) Even when the MC instruction is OFF, instructions from the MC instruction to the MCR instruction will be executed, so scan time will not be shortened.

POINT

If there are unnecessary contact instructions (FOR - NEXT, EI, DI, etc.) in ladders which use master controls, the CPU module will execute these instructions regardless of the ON/OFF state of the MC instruction.

(3) By changing the device designated by Ⓓ, the MC instruction can use the same nesting (N) number as often as desired.

(4) Coils from devices designated by Ⓓ are turned ON when the MC instruction is ON.
Further, using these same devices with the OUT instruction or other instructions will cause them to become double coils, so devices designated by Ⓓ should not be used within other instructions.

MCR

(1) This is the instruction for recovery from the master control, and indicates the end of the master control range of operation.
(2) Do not place contact instructions before the MCR instruction.

[Operation Errors]

(1) There are no operation errors associated with the MC or MCR instructions.

[Program Example]

The master control instruction can be used in nesting.

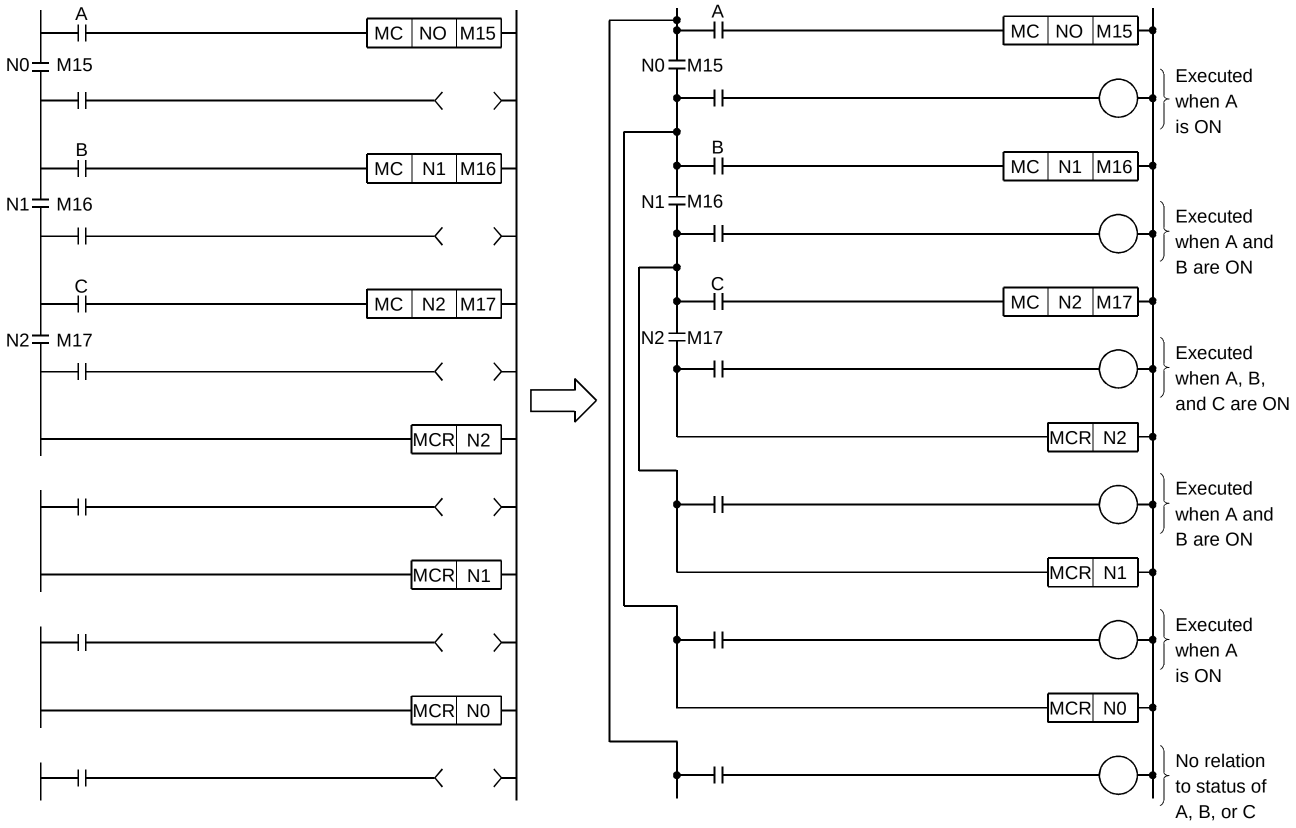
The different master control regions are distinguished by nesting (N).

Nesting can be performed from N0 to N14.

The use of nesting enables the creation of ladders which successively limit the execution condition of the program.

A ladder using nesting would appear as shown below:

[Ladder as displayed in the GPP ladder mode] [Ladder as it actually operates]


A
MC NO M15
N0 M15
B
MC N1 M16
N1 M16
C
MC N2 M17
N2 M17
MCR N2
MCR N1
MCR N0
Executed when A is ON
Executed when A and B are ON
Executed when A, B, and C are ON
Executed when A and B are ON
Executed when A is ON
No relation to status of A, B, or C

Cautions when Using Nesting Architecture

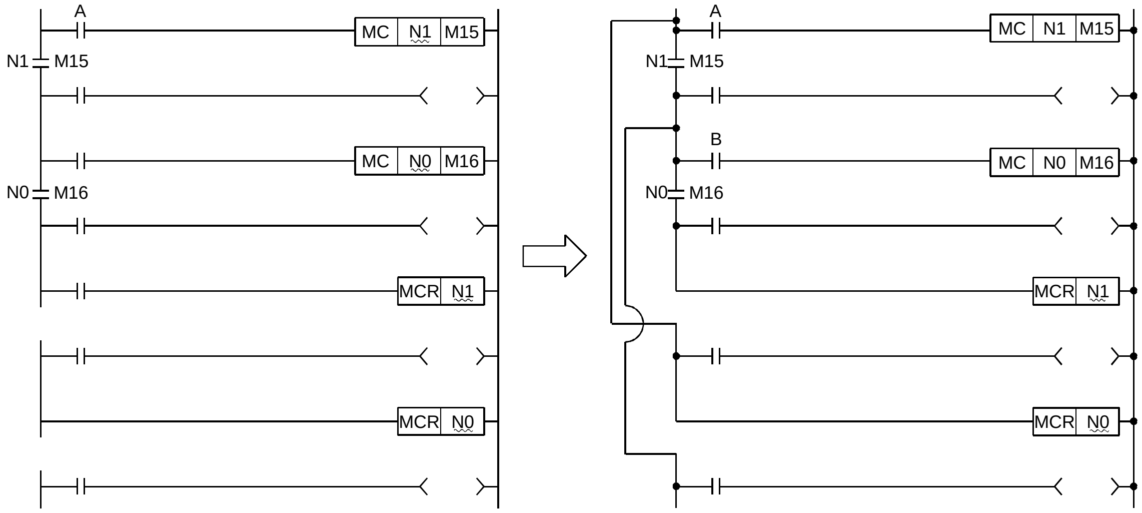
(1) Nesting can be used up to 15 times (N0 to N14)

When using nesting, nests should be inserted from the lower to higher nesting number (N) with the MC instruction, and from the higher to the lower order with the MCR instruction.

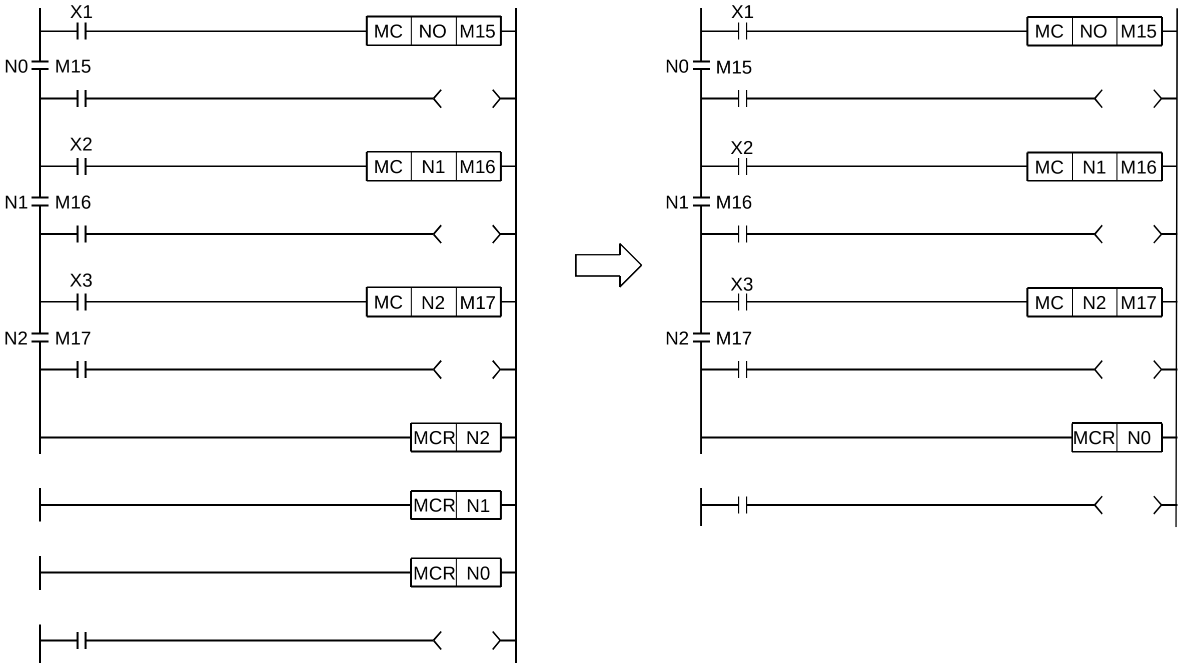
If this order is reversed, there will be no nesting architecture, and the QnACPU will not be capable of performing correct operations.

For example, if nesting is designated in the order N1 to N0 by the MC instruction, and also designated in the N1 to N0 order by the MCR instruction, the vertical bus will intersect and a correct master control ladder will not be produced.

[Ladder as displayed in the GPP ladder mode] [Ladder as it actually operates]



(2) If the nesting architecture results in MCR instructions concentrated in one location, all master controls can be terminated by use of just the lowest nesting number (N).



QCPU			QnA	Q4AR
PLC CPU		Process CPU		
Basic	High Performance			
○	○	○	○	○

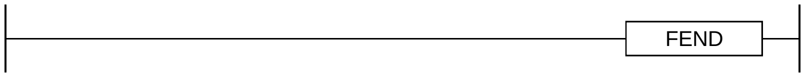
5.6 Termination Instructions

5.6.1 End main routine program (FEND)

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant	Other
	Bit	Word		Bit	Word				
—	—								

[Instruction Symbol] [Execution Condition]

FEND

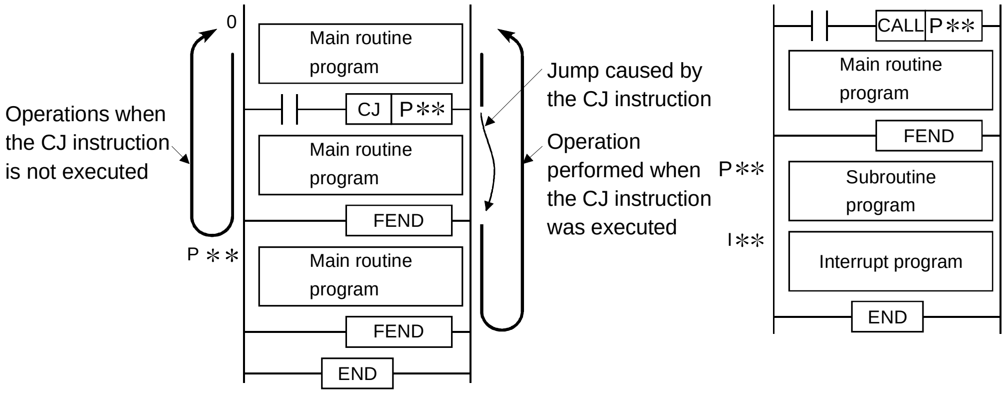


[Functions]

(1) The FEND instruction is used in cases where the CJ instruction or other instructions are used to cause a branch in the sequence program operations, and in cases where the main routine program is to be split from a subroutine program or an interrupt program.

(2) Execution of the FEND instruction will cause the CPU module to terminate the program it was executing.

(3) Even sequence programs following the FEND instruction can be displayed in ladder display at a peripheral devices. (Peripheral devices continue to display ladders until encountering an END instruction.)



(a) When using the CJ instruction

(b) When there is a subroutine or interrupt program

[Operation Errors]

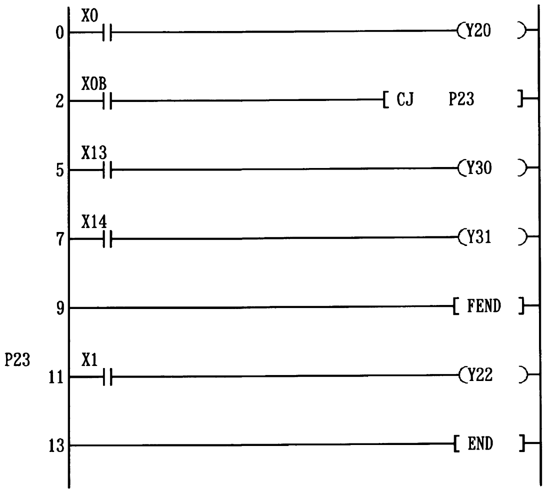
(1) In the following cases an operation error occurs, the error flag (SM0) turns ON, and an error code is stored at SD0.
- A FEND instruction is executed after the execution of a CALL, FCALL, ECALL, or EFCALL instruction, and before the execution of the RET instruction. (Error code: 4211)
- A FEND instruction is executed after the execution of a FOR instruction, and before the execution of a NEXT instruction. (Error code: 4200)

- A FEND instruction is executed during an interrupt program, and before the execution of an IRET instruction. (Error code: 4221)
- A FEND instruction is executed between the CHKCIR and CHKEND instructions. (Error code: 4230)
- A FEND instruction is executed between the IX and IXEND instructions. (Error code: 4231)

[Program Example]

(1) The following program uses the CJ instruction.

[Ladder Mode]



When XB is ON, jumps to label P23; from P23, executes the next step

Executed when XB is OFF

Indicates the termination of the sequence program when XB is OFF

[List Mode]

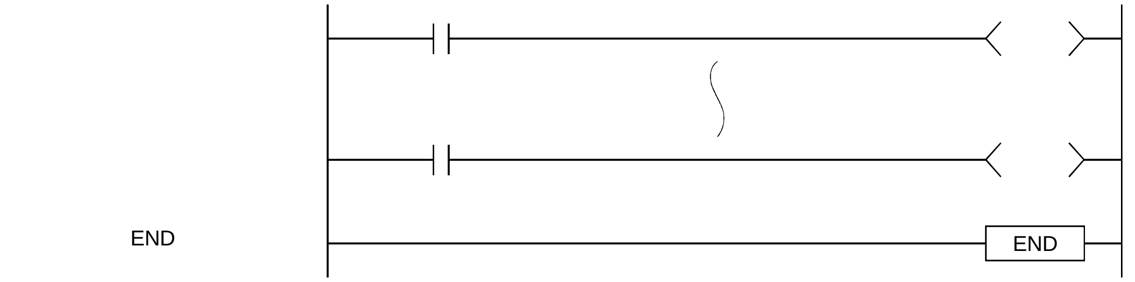
Steps	Instruction	Device
0	LD	X0
1	OUT	Y20
2	LD	X0B
3	CJ	P23
5	LD	X13
6	OUT	Y30
7	LD	X14
8	OUT	Y31
9	FEND	
10		P23
11	LD	X1
12	OUT	Y22
13	END	

QCPU			QnA	Q4AR
PLC CPU		Process CPU		
Basic	High Performance			
○	○	○	○	○

5.6.2 End sequence program (END)

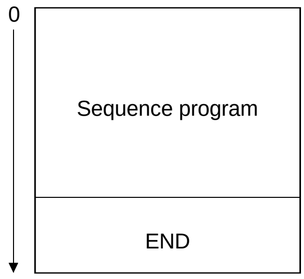
Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant K, H	Other
	Bit	Word		Bit	Word				U
—	—								

[Instruction Symbol] [Execution Condition]



[Functions]

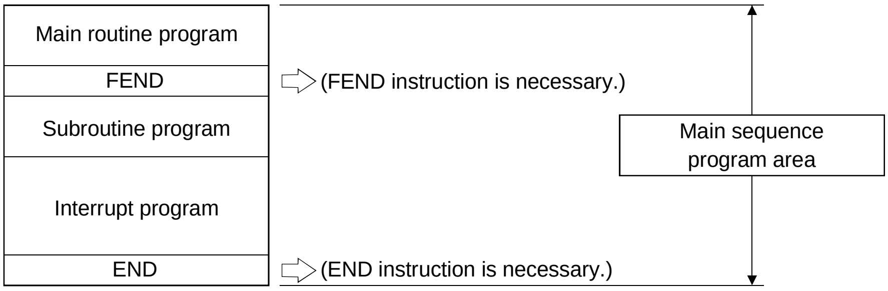
(1) Indicates termination of programs, including main routine program, subroutine program, and interrupt programs.
Execution of the END instruction will cause the CPU module to terminate the program that was being executed.



(2) An END instruction cannot be used during the execution of the main sequence program.
If it is necessary to perform END processing during the execution of a program, use the FEND instruction.

(3) When programming in the ladder mode of a peripheral device, it is not necessary to input an END instruction.

(4) The use of the END and FEND instructions is broken down as follows for main routine programs, subroutine programs, and interrupt programs:


Main routine program
FEND
(FEND instruction is necessary.)
Subroutine program
Main sequence program area
Interrupt program
END
(END instruction is necessary.)

[Operation Errors]

(1) In the following cases an operation error occurs, the error flag (SM0) turns ON, and an error code is stored at SD0.

- An END instruction was executed before the execution of the RET instruction and after the execution of the CALL, FCALL, ECALL, or EFCALL instruction. (Error code: 4211)
- An END instruction was executed before the execution of a NEXT instruction and after the execution of the FOR instruction. (Error code: 4200)
- An END instruction was executed during an interrupt program prior to the execution of the IRET instruction. (Error code: 4221)
- An END instruction was executed within the CHKCIR to CHKEND instruction loop. (Error code: 4230)
- An END instruction was executed within the IX to IXEND instruction loop. (Error code: 4231)

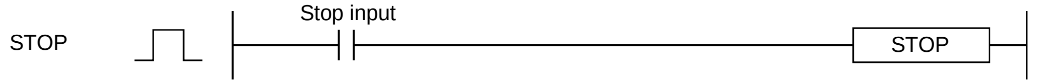
QCPU			QnA	Q4AR
PLC CPU		Process CPU		
Basic	High Performance			
○	○	○	○	○

5.7 Other Instructions

5.7.1 Sequence program stop (STOP)

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant K, H	Other
	Bit	Word		Bit	Word				U
—	—								

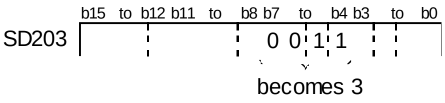
[Instruction Symbol] [Execution Condition]



[Functions]

(1) When stop input is turned ON, output Y is reset and the CPU module operations are terminated.
(The same result will take place if the RUN/STOP (key) switch is turned to the STOP setting.)

(2) Execution of the STOP instruction will cause the value of b4 to b7 of the special register SD203 to become "3".



(3) In order to restart CPU module operations after the execution of the STOP instruction, return the RUN/STOP key switch, which has been changed from RUN to STOP, back to the RUN position.

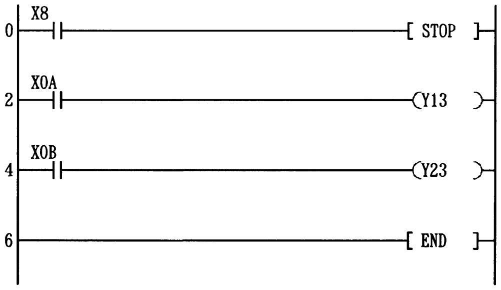
[Operation Errors]

(1) In the following cases an operation error occurs, the error flag (SM0) turns ON, and an error code is stored at SD0.
- A STOP instruction was executed before the execution of the RET instruction and after the execution of the CALL, ECALL instruction. (Error code: 4211)
- A STOP instruction was executed before the execution of a NEXT instruction and after the execution of the FOR instruction. (Error code: 4200)
- A STOP instruction was executed during an interrupt program prior to the execution of the IRET instruction. (Error code: 4221)
- A STOP instruction was executed within the CHKCIR to CHKEND instruction loop. (Error code: 4230)
- A STOP instruction was executed within the IX to IXEND instruction loop. (Error code: 4231)

[Program Example]

(1) The following program stops the CPU module when X8 goes ON

[Ladder Mode]



Causes programmable controller to stop when X8 goes ON.

Sequence program

[List Mode]

Steps	Instruction	Device
0	LD	X8
1	STOP	
2	LD	X0A
3	OUT	Y13
4	LD	X0B
5	OUT	Y23
6	END	

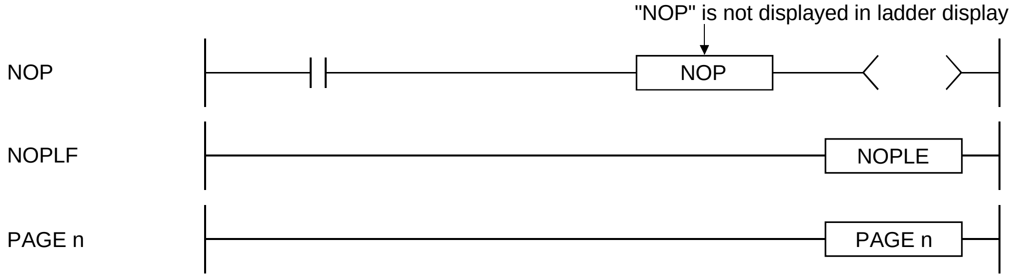
QCPU			QnA	Q4AR
PLC CPU		Process CPU		
Basic	High Performance			
○	○	○	○	○

5.7.2 No operation (NOP, NOPLF, PAGE n)

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant K, H	Other
	Bit	Word		Bit	Word				
—	—								

[Instruction Symbol]   [Execution Condition]

"NOP" is not displayed in ladder display.



[Functions]

NOP

(1) This is a no operation instruction that has no impact on any operations up to that point.

(2) The NOP instruction is used in the following cases:

(a) To insert space for sequence program debugging.

(b) To delete an instruction without having to change the number of steps. (Replace the instruction with NOP)

(c) To temporarily delete an instruction.

NOPLF

(1) This is a no operation instruction that has no impact on any operations up to that point.

(2) The NOPLF instruction is used when printing from a peripheral device to force a page change at any desired location.

(a) When printing ladders

- A page break will be inserted between ladder blocks with the presence of the NOPLF instruction.
- The ladder cannot be displayed correctly if an NOPLF instruction is inserted in the midst of a ladder block.
  Do not insert an NOPLF instruction in the midst of a ladder block.

(b) When printing instruction lists

- The page will be changed after the printing of the NOPLF instruction.

(3) Refer to the Operating Manual for the peripheral device in use for details of printouts from peripheral devices.

PAGE n

(1) This is a no operation instruction that has no impact on any operations up to that point.

(2) Causes processing from step 0 of the designated nth page of the program following the PAGE n instruction. (Peripheral device display, printers, etc.)

(3) If there is no PAGE n instruction, processing begins from page 0.

[Operation Errors]

(1) There are no errors associated with the NOP, NOPLF, or PAGE instructions.

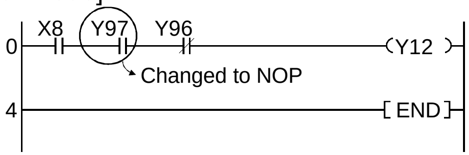
[Program Example]

NOP

(1) Contact closed .... Deletes AND or ANI instruction

Before change

[Ladder Mode]

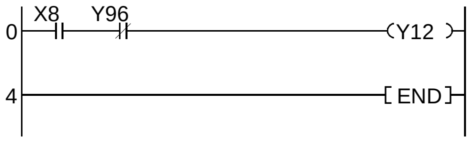


[List Mode]

Steps	Instruction	Device
0	LD	X8
1	AND	Y97
2	ANI	Y96
3	OUT	Y12
4	END	

After change

[Ladder Mode]



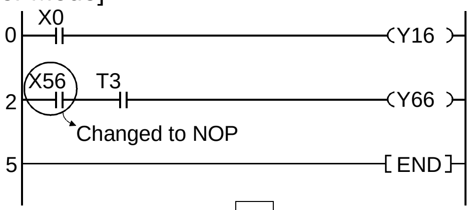
[List Mode]

Steps	Instruction	Device
0	LD	X8
1	NOP	
2	ANI	Y96
3	OUT	Y12
4	END	

(2) Contact closed .... LD, LDI changed to NOP (Note carefully that changing the LD and LDI instructions to NOP completely changes the nature of the ladder.)

Before change

[Ladder Mode]

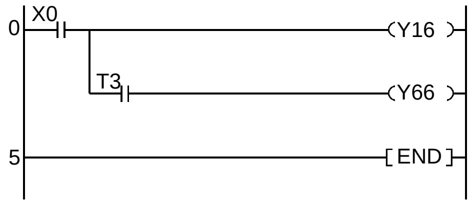


[List Mode]

Steps	Instruction	Device
0	LD	X0
1	OUT	Y16
2	LD	X56
3	AND	T3
4	OUT	Y66
5	END	

After change

[Ladder Mode]

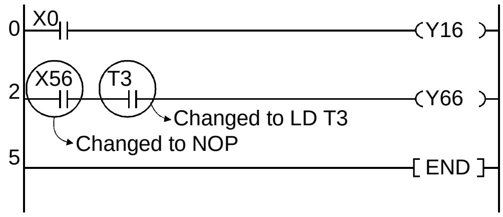


[List Mode]

Steps	Instruction	Device
0	LD	X0
1	OUT	Y16
2	NOT	
3	AND	T3
4	OUT	Y66
5	END	

Before change

[Ladder Mode]

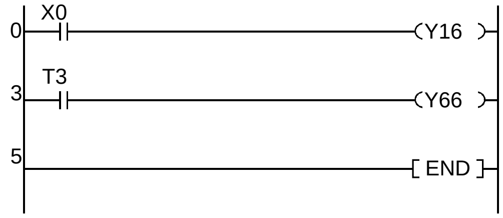


[List Mode]

Steps	Instruction	Device
0	LD	X0
1	OUT	Y16
2	LD	X56
3	AND	T3
4	OUT	Y66
5	END	

After change

[Ladder Mode]

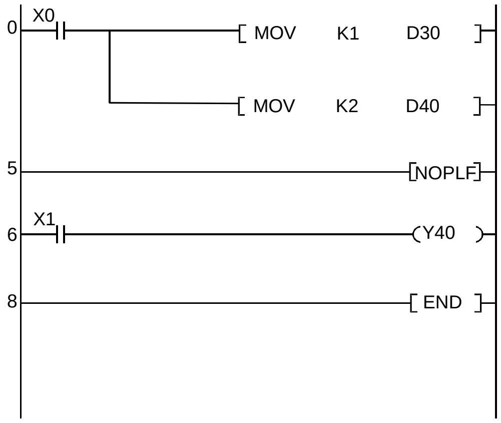


[List Mode]

Steps	Instruction	Device
0	LD	X0
1	OUT	Y16
2	NOP	
3	LD	T3
4	OUT	Y66
5	END	

NOPLF

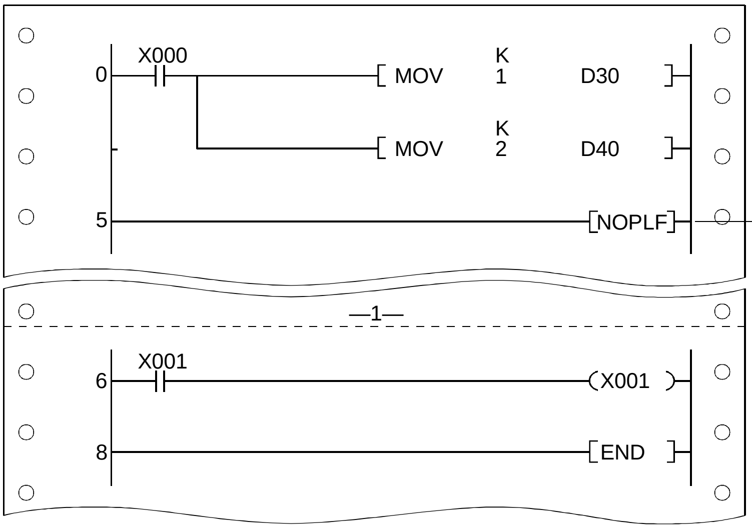
[Ladder Mode]



[List Mode]

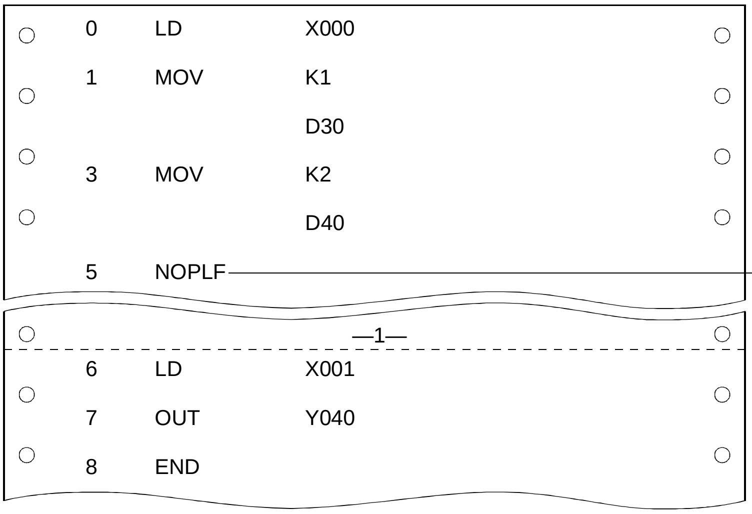
Steps	Instruction	Device
0	LD	X0
1	MOV	K1 D30
3	MOV	K2 D40
5	NOPLF	
6	LD	X1
7	OUT	Y40
8	END	

- Printing the ladder will result in the following:



Page change forced when NOPLF is inserted between two ladder blocks.

- Printing an instruction list with the NOPLF instruction will result in the following:

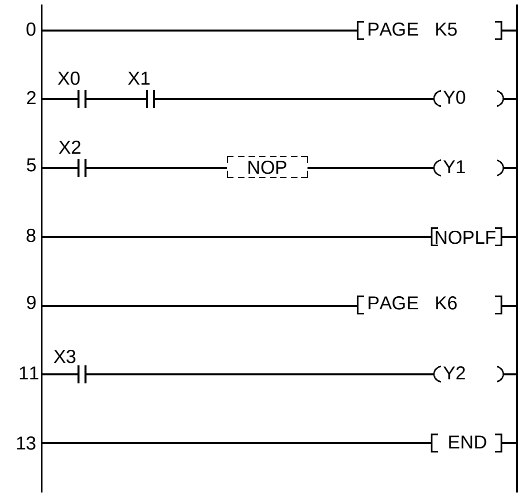


```
0       LD          X000
1       MOV         K1
                    D30
3       MOV         K2
                    D40
5       NOPLF
                 —1—
6       LD          X001
7       OUT         Y040
8       END
```

Changes pages after printing NOPLF

PAGE n

[Ladder Mode]



[List Mode]

Steps	Instruction	Device
0	PAGE	K5
2	LD	X0
3	AND	X1
4	OUT	Y0
5	LD	X2
6	NOP	
7	OUT	Y1
8	NOPLF	
9	PAGE	K6
11	LD	X3
12	OUT	Y2
13	END	

6. BASIC INSTRUCTIONS

The following types of basic instructions are available.

Instruction	Meaning	Reference Section
Comparison operation instruction	Compare data to data	Chapter 6.1
Arithmetic operation instructions	Adds, subtracts multiplies, divides, increments, or decrements data with other data	Chapter 6.2
Data conversion instruction	Converts data types	Chapter 6.3
Data transfer instruction	Transmits designated data	Chapter 6.4
Program branch instruction	Program jumps	Chapter 6.5
Program execution control instructions	Enables and disables program interrupts	Chapter 6.6
Refresh instruction	Refreshes bit devices	Chapter 6.7

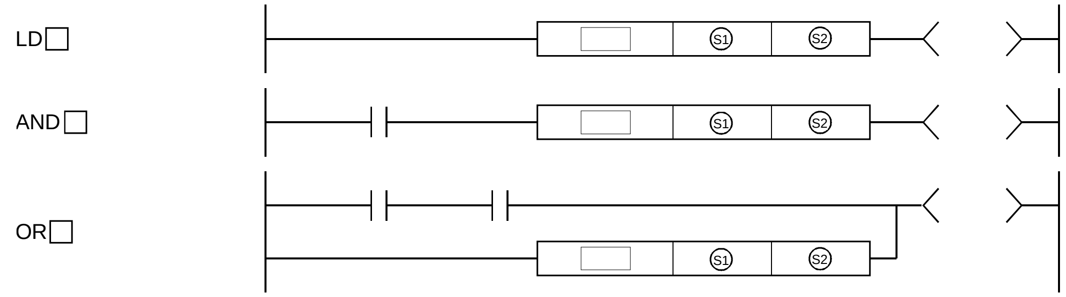
QCPU			QnA	Q4AR
PLC CPU		Process CPU		
Basic	High Performance			
○	○	○	○	○

6.1 Comparison Operation Instruction

6.1.1 BIN 16-bit data comparisons (=, < >, >, <=, <, >=)

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant K, H	Other
	Bit	Word		Bit	Word				
Ⓢ1	○								–
Ⓢ2	○								–

[Instruction Symbol] [Execution Condition] □ indicates the signs =, < >, >, <=, <, or >=



[Set Data]

Set Data	Meaning	Data Type
Ⓢ1	Comparative data, or device number where comparative data is stored	BIN 16 bits
Ⓢ2		

[Functions]

(1) Treats BIN 16-bit data from device designated by Ⓢ1 and BIN 16-bit data from device designated by Ⓢ2 as an A contact, and performs comparison operation.

(2) The results of the comparison operations for the individual instructions are as follows:

Instruction Symbol in □	Condition	Comparison Operation Result	Instruction Symbol in □	Condition	Comparison Operation Result
=	Ⓢ1 = Ⓢ2	Continuity	=	Ⓢ1 ≠ Ⓢ2	Non-continuity
< >	Ⓢ1 ≠ Ⓢ2		< >	Ⓢ1 = Ⓢ2	
>	Ⓢ1 > Ⓢ2		>	Ⓢ1 ≤ Ⓢ2	
< =	Ⓢ1 ≤ Ⓢ2		< =	Ⓢ1 > Ⓢ2	
<	Ⓢ1 < Ⓢ2		<	Ⓢ1 ≥ Ⓢ2	
> =	Ⓢ1 ≥ Ⓢ2		> =	Ⓢ1 < Ⓢ2	

(3) In cases where hexadecimal constants have been designated by Ⓢ1 and Ⓢ2, or when a numerical value (8 to F) where the highest bit (b15) will be 1 has been designated, the value will be read as a negative BIN value number for purposes of the comparison.

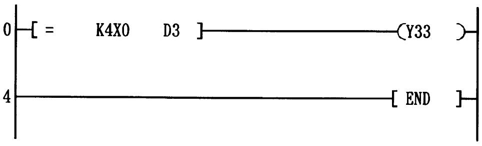
[Operation Errors]

(1) There are no operation errors associated with the =, <>, >, <=, <, or >= instructions.

[Program Example]

(1) The following program compares the data at X0 to XF with the data at D3, and turns Y33 ON if the data is identical.

[Ladder Mode]

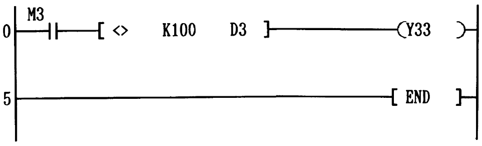


[List Mode]

Steps	Instruction	Device
0	LD=	K4X0 D3
3	OUT	Y33
4	END	

(2) The following program compares BIN value K100 to the data at D3, and establishes continuity if the data in D3 is something other than 100.

[Ladder Mode]

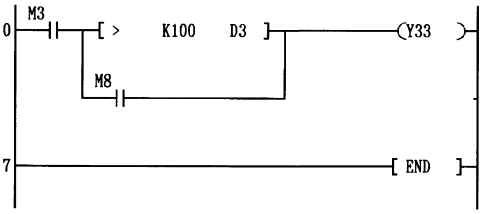


[List Mode]

Steps	Instruction	Device
0	LD	M3
1	AND<>	K100 D3
4	OUT	Y33
5	END	

(3) The following program compares the BIN value 100 with the data in X0 to XF, and establishes continuity if the D3 data is less than 100.

[Ladder Mode]

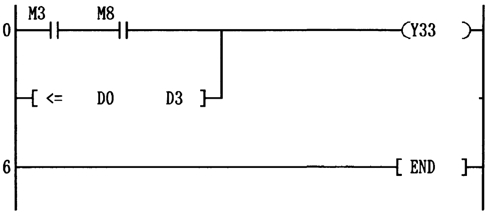


[List Mode]

Steps	Instruction	Device
0	LD	M3
1	LD>	K100 D3
4	OR	M8
5	ANB	
6	OUT	Y33
7	END	

(4) The following program compares the data in D0 and D3, and if the data in D0 is equal to or less than the data in D3, establishes continuity.

[Ladder Mode]



[List Mode]

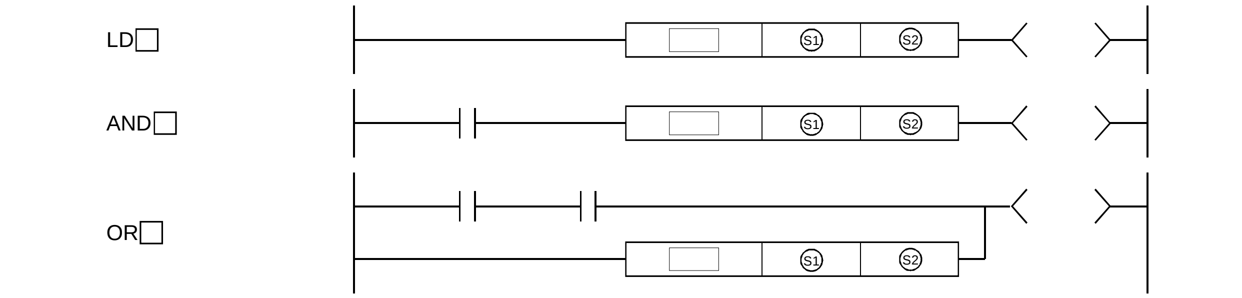
Steps	Instruction	Device
0	LD	M3
1	AND	M8
2	OR<=	D0 D3
5	OUT	Y33
6	END	

QCPU			QnA	Q4AR
PLC CPU		Process CPU		
Basic	High Performance			
○	○	○	○	○

6.1.2 BIN 32-bit data comparisons (D=, D< >, D>, D<=,D<, D>=)

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant K, H	Other
	Bit	Word		Bit	Word				
(S1)	○								—
(S2)	○								—

[Instruction Symbol] [Execution Condition] □ indicates the signs D=, D< >, D>, D<=,D<, or D>=



[Set Data]

Set Data	Meaning	Data Type
(S1)	Comparative data, or device number where comparative data is stored	BIN 32 bits
(S2)		

[Functions]

(1) Treats BIN 32-bit data from device designated by (S1) and BIN 32-bit data from device designated by (S2) as an A contact, and performs comparison operation.

(2)The results of the comparison operations for the individual instructions are as follows:

Instruction Symbol in □	Condition	Comparison Operation Result	Instruction Symbol in □	Condition	Comparison Operation Result
D =	(S1) = (S2)	Continuity	D =	(S1) ≠ (S2)	Non-continuity
D < >	(S1) ≠ (S2)		D < >	(S1) = (S2)	
D >	(S1) > (S2)		D >	(S1) ≤ (S2)	
D < =	(S1) ≤ (S2)		D < =	(S1) > (S2)	
D <	(S1) < (S2)		D <	(S1) ≥ (S2)	
D > =	(S1) ≥ (S2)		D > =	(S1) < (S2)	

(3) In cases where hexadecimal constants have been designated by (S1) and (S2), or when a numerical value (8 to F) where the highest bit (b31) will be 1 has been designated, the value will be read as a negative BIN value number for the purposes of the comparison.

(4) Data used for comparison should be designated by a 32-bit instruction (DMOV instruction, etc.).
If designation is made with a 16-bit instruction (MOV instruction, etc.), comparisons of large and small values cannot be performed correctly.

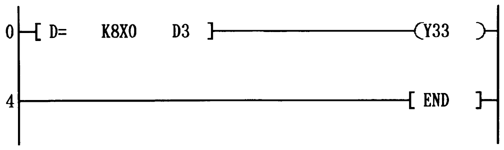
[Operation Errors]

(1) There are no operation errors associated with the =, <>, >, <=, <, or >= instructions.

[Program Example]

(1) The following program compares the data at X0 to XF with the data at D3, and turns Y33 ON if the data is identical.

[Ladder Mode]

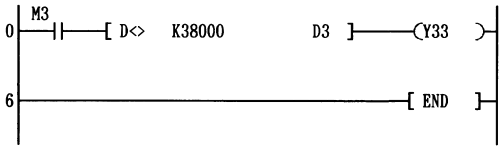


[List Mode]

Steps	Instruction	Device
0	LDD=	K8X0
		D3
3	OUT	Y33
4	END	

(2) The following program compares BIN value K38000 to the data at D3, and D4, and establishes continuity if the data in D3 and D4 is something other than 38000.

[Ladder Mode]

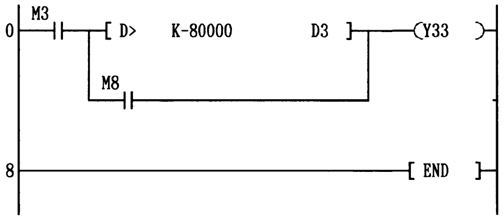


[List Mode]

Steps	Instruction	Device
0	LD	M3
1	ANDD<>	K38000
		D3
5	OUT	Y33
6	END	

(3) The following program compares BIN value K-80000 to the data at D3 and D4, and establishes continuity if the data in D3 and D4 is less than -80000.

[Ladder Mode]

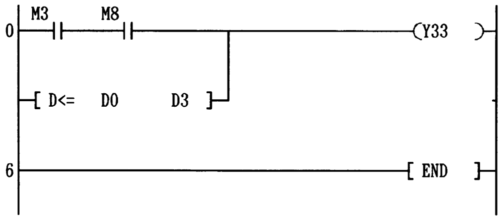


[List Mode]

Steps	Instruction	Device
0	LD	M3
1	LDD>	K-80000
		D3
5	OR	M8
6	ANB	
7	OUT	Y33
8	END	

(4) The following program compares the data in D0 and D1 with the data in D3 and D4, and establishes continuity if the data in D0 and D1 is equal to or less than the data in D3 and D4.

[Ladder Mode]



[List Mode]

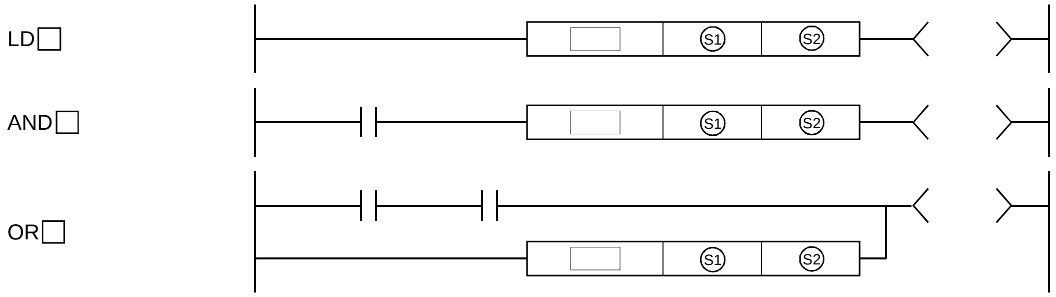
Steps	Instruction	Device
0	LD	M3
1	AND	M8
2	ORD<=	D0
		D3
5	OUT	Y33
6	END	

QCPU			QnA	Q4AR
PLC CPU		Process CPU		
Basic	High Performance			
×	○	○	○	○

6.1.3 Floating decimal point data comparisons (E=, E< >, E>, E<=, E<, E>=)

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant E	Other
	Bit	Word		Bit	Word				
(S1)	—	○		—	○		—	○	—
(S2)	—	○		—	○		—	○	—

[Instruction Symbol] [Execution Condition] □ indicates the signs E=, E< >, E>, E<=, E<, or E>=



[Set Data]

Set Data	Meaning	Data Type
(S1)	Comparative data, or device number where comparative data is stored	Real number
(S2)		

[Functions]

(1) The floating decimal point data from device designated by (S1) and floating decimal point data from device designated by (S2) as A contact, and performs comparison operation.

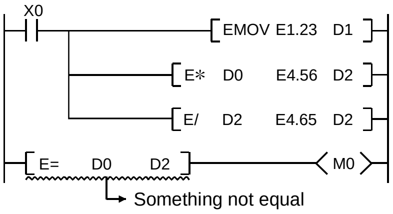
(2)The results of the comparison operations for the individual instructions are as follows.

Instruction Symbol in □	Condition	Comparison Operation Result	Instruction Symbol in □	Condition	Comparison Operation Result
E =	(S1) = (S2)	Continuity	E =	(S1) ≠ (S2)	Non-continuity
E < >	(S1) ≠ (S2)		E < >	(S1) = (S2)	
E >	(S1) > (S2)		E >	(S1) ≤ (S2)	
E < =	(S1) ≤ (S2)		E < =	(S1) > (S2)	
E <	(S1) < (S2)		E <	(S1) ≥ (S2)	
E > =	(S1) ≥ (S2)		E > =	(S1) < (S2)	

POINT

Note that use of the = instruction can on occasion result in situations where errors cause the two values to not be equal.

Example



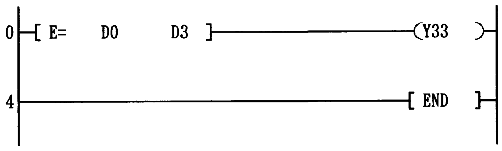
[Operation Errors]

(1) There are no operation errors associated with the E=, E< >, E>, E<=, E<, or, E>= instructions.

[Program Example]

(1) The following program compares floating decimal point real number data at D0 and D1 to floating decimal point real number data at D3 and D4.

[Ladder Mode]

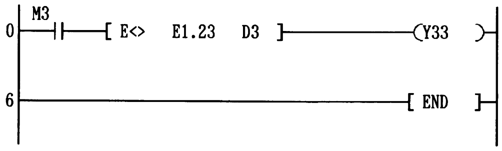


[List Mode]

Steps	Instruction	Device
0	LDE=	D0 D3
3	OUT	Y33
4	END	

(2) The following program compares the floating decimal point real number 1.23 to the floating decimal point real number data at D3 and D4.

[Ladder Mode]

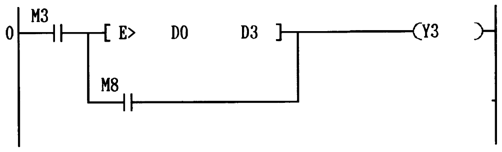


[List Mode]

Steps	Instruction	Device
0	LD	M3
1	ANDE<>	E1.23 D3
5	OUT	Y33
6	END	

(3) The following program compares floating decimal point real number data at D0 and D1 to floating decimal point real number data at D3 and D4.

[Ladder Mode]

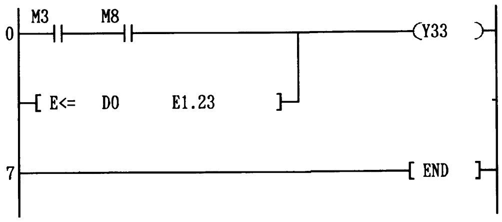


[List Mode]

Steps	Instruction	Device
0	LD	M3
1	LDE>	D0 D3
4	OR	M8
5	ANB	
6	OUT	Y3
7	END	

(4) The following program compares the floating decimal point data at D0 and D1 to the floating decimal point real number 1.23.

[Ladder Mode]



[List Mode]

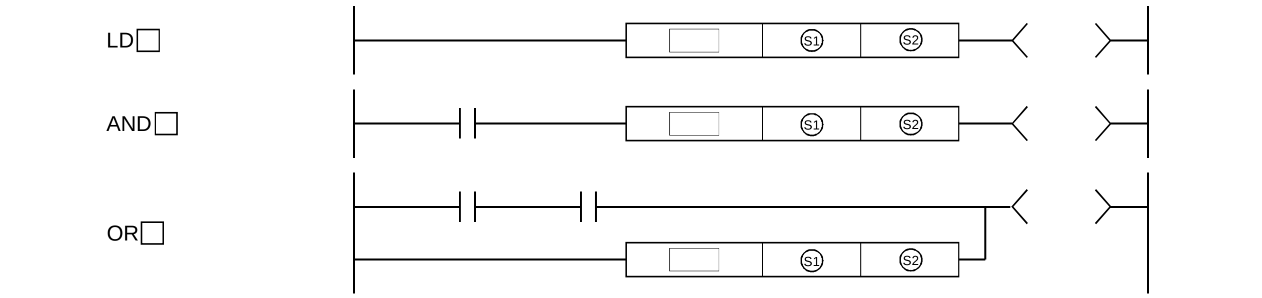
Steps	Instruction	Device
0	LD	M3
1	AND	M8
2	ORE<=	D0 E1.23
6	OUT	Y33
7	END	

QCPU				QnA	Q4AR
PLC CPU		Process CPU			
Basic	High Performance				
×	○	○		○	○

6.1.4 Character string data comparisons ($=, $< >, $>, $<=, $<, $>=)

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant $	Other
	Bit	Word		Bit	Word				
S1	—	○		—				○	—
S2	—	○		—				○	—

[Instruction Symbol] [Execution Condition] □ indicates the signs $=, $< >, $>, $<=, $<, or $>=



[Set Data]

Set Data	Meaning	Data Type
S1 S2	First number of comparison data, or of the device where comparison data is being stored.	Character string

[Functions]

(1) Treats character string data stored following the device designated by S1 and character string data stored following the device designated by S2 as A contact, and performs comparison operation.

(2) A comparison operation involves the character-by-character comparison of the ASCII code of the first character in the character string.

(3) The S1 and S2 character strings encompass all characters from the designated device number to the next device number storing the code "00H".

(a) If all character strings match, the comparison result will be matched.

	b15- - - -b8	b7- - - - b0
S1	42H (B)	41H (A)
S1+1	44H (D)	43H (C)
S1+2	00H	45H (E)

"ABCDE"

□

	b15- - - -b8	b7- - - - b0
S2	42H (B)	41H (A)
S2+1	44H (D)	43H (C)
S2+2	00H	45H (E)

"ABCDE"

Instruction Symbol in □	Comparison Operation Result	Instruction Symbol in □	Comparison Operation Result
$ =	Continuity	$< =	Continuity
$< >	Non-continuity	$<	Non-continuity
$>	Non-continuity	$> =	Continuity

(b) If the character strings are different, the character string with the larger character code will be the larger.

	b15- - - -b8	b7- - - - b0			b15- - - -b8	b7- - - - b0
S1	42H (B)	41H (A)		S2	42H (B)	41H (A)
S1+1	44H (D)	43H (C)	□	S2+1	44H (D)	43H (C)
S1+2	00H	46H (F)		S2+2	00H	45H (E)
	"ABCDF"				"ABCDE"	

Instruction Symbol in □	Comparison Operation Result	Instruction Symbol in □	Comparison Operation Result
$ =	Non-continuity	$< =	Non-continuity
$< >	Continuity	$<	Non-continuity
$>	Continuity	$> =	Continuity

(c) If the character strings are different, the first different sized character code will determine whether the character string is larger or smaller.

	b15- - - -b8	b7- - - - b0			b15- - - -b8	b7- - - - b0
S1	32H (2)	31H (1)		S2	32H (2)	31H (1)
S1+1	33H (3)	34H (4)	□	S2+1	34H (4)	33H (3)
S1+2	00H	35H (5)		S2+2	00H	35H (5)
	"12435"				"12345"	

Instruction Symbol in □	Comparison Operation Result	Instruction Symbol in □	Comparison Operation Result
$ =	Non-continuity	$< =	Non-continuity
$< >	Continuity	$<	Non-continuity
$>	Continuity	$> =	Continuity

(4) If the character strings designated by S1 and S2 are of different lengths, the data with the longer character string will be larger.

	b15- - - -b8	b7- - - - b0			b15- - - -b8	b7- - - - b0
S1	32H (2)	31H (1)		S2	32H (2)	31H (1)
S1+1	34H (4)	33H (3)	□	S2+1	34H (4)	33H (3)
S1+2	36H (6)	35H (5)		S2+2	36H (6)	35H (5)
S1+3	00H	37H (7)		S2+3	00H	00H
	"1234567"				"123456"	

Instruction Symbol in □	Comparison Operation Result	Instruction Symbol in □	Comparison Operation Result
$ =	Non-continuity	$< =	Continuity
$< >	Continuity	$<	Continuity
$>	Non-continuity	$> =	Non-continuity

[Operation Errors]

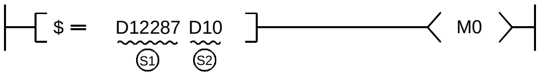
(1) In the following cases an operation error occurs, the error flag (SM0) turns ON, and an error code is stored at SD0.

- The code “00H” or non-matching does not exist within the relevant device range following the device number designated by S1 or S2. (Error code: 4101)

POINT

At the same time that it is conducting a character string comparison, character string data comparison instruction also checks the device range.
For this reason, even in cases where the character string exceeds the device range, the character string data is compared. If character non-matching is detected within the device range at this time, the comparison operation results are output without returning an operation error.

Example



S1 data

D12287	"B"	"A"
W0	00H	"C"

S2 data

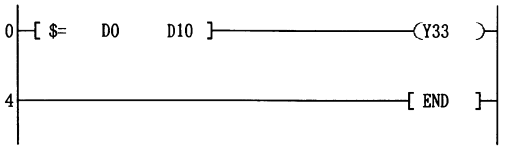
D10	"Z"	"A"
D11	00H	"C"

＊ In the example shown above, the S1 character string exceeds the device range, but because its second character is different from that of S2, the comparison result is "S1 does not equal S2", and the operation result is non-continuity.
In this case, because the non-continuity detection is for D12287 (inside the device range), there will be no operation error returned.

[Program Example]

(1) The following program compares character strings stored following D0 and characters following D10.

[Ladder Mode]

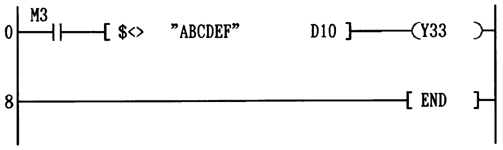


[List Mode]

Steps	Instruction	Device
0	LD$=	D0 D10
3	OUT	Y33
4	END	

(2) The following program compares the character string "ABCDEF" with the character string stored following D10.

[Ladder Mode]

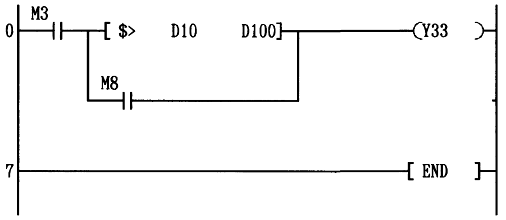


[List Mode]

Steps	Instruction	Device
0	LD	M3
1	AND$<>	"ABCDEF" D10
7	OUT	Y33
8	END	

(3) The following program compares the character string stored following D10 with the character string stored following D100.

[Ladder Mode]

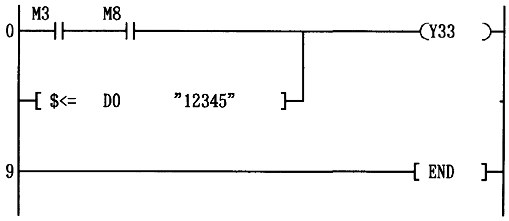


[List Mode]

Steps	Instruction	Device
0	LD	M3
1	LD$>	D10 D100
4	OR	M8
5	ANB	
6	OUT	Y33
7	END	

(4) The following program compares the character string stored following D10 with the character string "12345."

[Ladder Mode]



[List Mode]

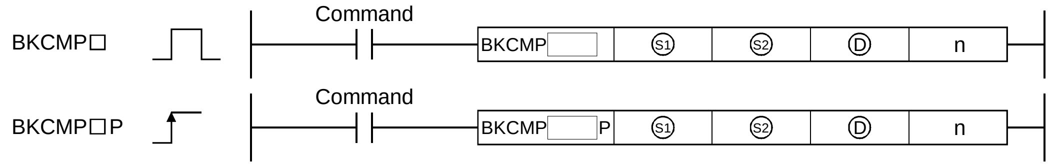
Steps	Instruction	Device
0	LD	M3
1	AND	M8
2	OR$<=	D0 "12345"
8	OUT	Y33
9	END	

QCPU			QnA	Q4AR
PLC CPU		Process CPU		
Basic	High Performance			
○	○	○	○	○

6.1.5 BIN block data comparisons (BKCMP, BKCMP□P)

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant K, H	Other
	Bit	Word		Bit	Word				
Ⓢ1	–	○		–				○	–
Ⓢ2	–	○		–				–	–
Ⓓ	○	○		–				–	–
n	○	○		○				○	–

[Instruction Symbol] [Execution Condition] □ indicates the signs =, < >, >, <=, <, or >=



[Set Data]

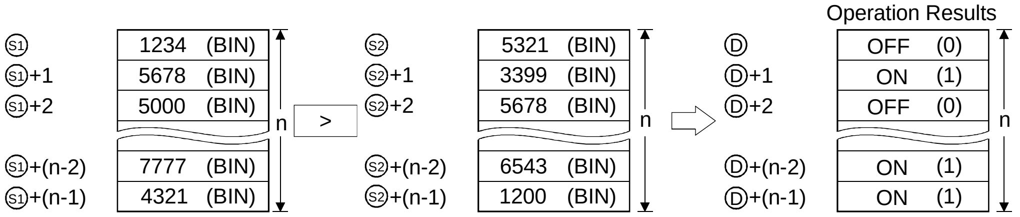
Set Data	Meaning	Data Type
Ⓢ1	Data being compared, or first number of the device where the data being compared is being stored	BIN 16 bits
Ⓢ2	First number of the device where the comparison data is being stored	BIN 16 bits
Ⓓ	First number of the device where the results of the comparison operation are being stored	Bit
n	Number of data blocks compared	BIN 16 bits

[Functions]

(1) Compares BIN 16-bit data the nth point from the device number designated by Ⓢ1 with BIN 16-bit data the nth point from the device number designated by Ⓢ2, and stores the result from the device designated by Ⓓ onward.

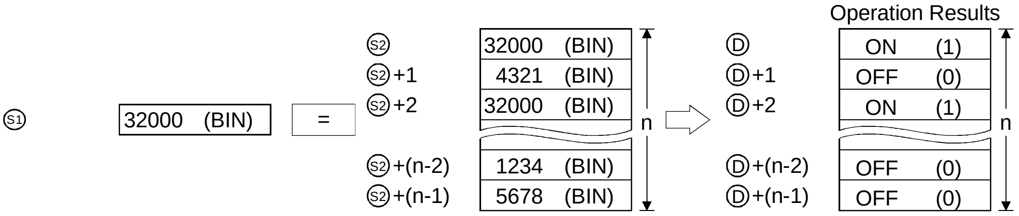
(a) If the comparison condition has been met, the device designated by Ⓓ will be turned ON.

(b) If the comparison condition has not been met, the device designated by Ⓓ will be turned OFF.



(2) The comparison operation is conducted in 16-bit units.

(3) The constant designated by Ⓢ1 can be between -32768 and 32767 (BIN 16-bit data).



(4) The results of the comparison operations for the individual instructions are as follows:

Instruction Symbols	Condition	Comparison Operation Result	Instruction Symbols	Condition	Comparison Operation Result
BKCMP =	Ⓢ1 = Ⓢ2	ON (1)	BKCMP =	Ⓢ1 ≠ Ⓢ2	OFF (0)
BKCMP < >	Ⓢ1 ≠ Ⓢ2		BKCMP < >	Ⓢ1 = Ⓢ2	
BKCMP >	Ⓢ1 > Ⓢ2		BKCMP >	Ⓢ1 ≤ Ⓢ2	
BKCMP < =	Ⓢ1 ≤ Ⓢ2		BKCMP < =	Ⓢ1 > Ⓢ2	
BKCMP <	Ⓢ1 < Ⓢ2		BKCMP <	Ⓢ1 ≥ Ⓢ2	
BKCMP > =	Ⓢ1 ≥ Ⓢ2		BKCMP > =	Ⓢ1 < Ⓢ2	

(5) If all comparison results stored n-points from Ⓓ are ON (1), SM704 (block comparison signal) goes ON.

[Operation Errors]

(1) In the following cases an operation error occurs, the error flag (SM0) turns ON, and an error code is stored at SD0.

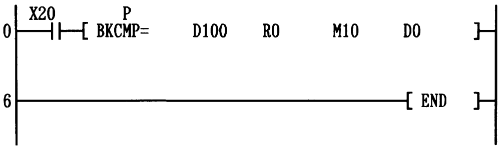
- The range of the device n points from a device designated by Ⓢ1, Ⓢ2 or Ⓓ exceeds the relevant device. (Error code: 4101)
- The device range for n points starting from the device designated by Ⓢ1 overlaps with the device range for n points starting from the device designated by Ⓓ. (Error code: 4101)
- The device range for n points starting from the device designated by Ⓢ2 overlaps with the device range for n points starting from the device designated by Ⓓ. (Error code: 4101)
- The device range for n points starting from the device designated by Ⓢ1 overlaps with the device range for n points starting from the device designated by Ⓢ2. (Error code: 4101)

(2) See Section 3.6 for information regarding errors during index modification.

[Program Example]

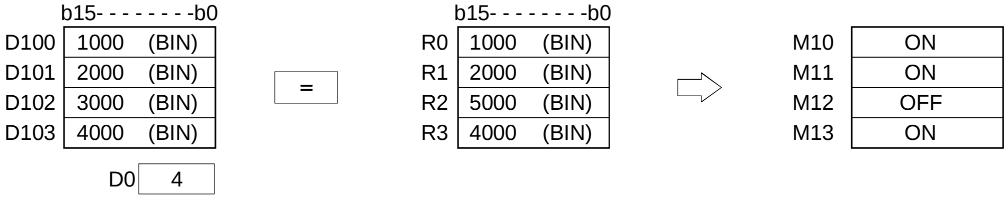
(1) The following program performs a comparison operation when X20 goes ON, comparing the data for the number of points from D100 equivalent to the value stored in D0 with the data the number of points from R0 equivalent to the value stored in D0, and stores the result from M10 onward.

[Ladder Mode]



[List Mode]

Steps	Instruction	Device
0	LD	X20
1	BKCMP=P	D100 R0 M10 D0
6	END	



	b15- - - - - - - -b0
D100	1000 (BIN)
D101	2000 (BIN)
D102	3000 (BIN)
D103	4000 (BIN)

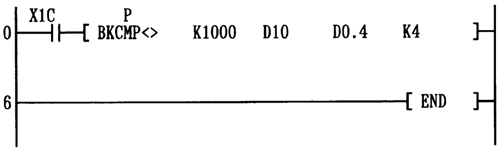
D0 4

	b15- - - - - - - -b0
R0	1000 (BIN)
R1	2000 (BIN)
R2	5000 (BIN)
R3	4000 (BIN)

M10	ON
M11	ON
M12	OFF
M13	ON

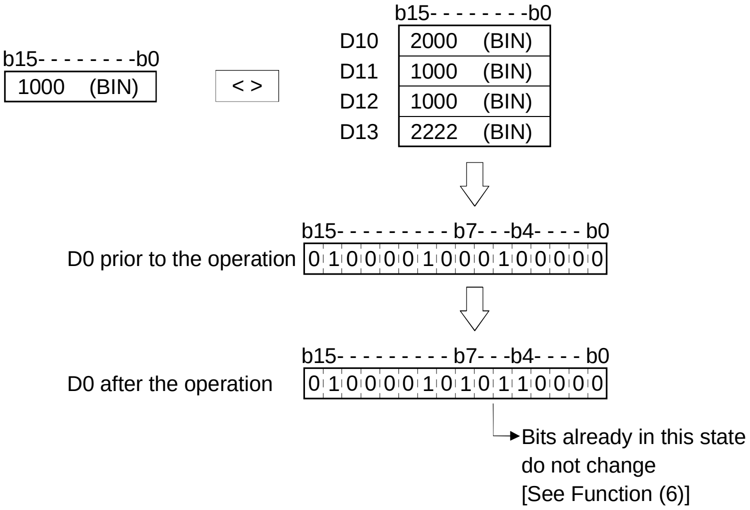
(2) The following program performs a comparison operation when X1C goes ON, comparing the constant K1000 with the data 4 points from D10, and stores the result in b4 to b7 of D0.

[Ladder Mode]



[List Mode]

Steps	Instruction	Device
0	LD	X1C
1	BKCMP<>P	K1000 D10 D0.4 K4
6		END



b15- - - - - - - -b0
1000 (BIN)
< >
b15- - - - - - - - - b7- - -b4- - - - b0
b15- - - - - - - - - b7- - -b4- - - - b0

	b15- - - - - - - -b0
D10	2000 (BIN)
D11	1000 (BIN)
D12	1000 (BIN)
D13	2222 (BIN)

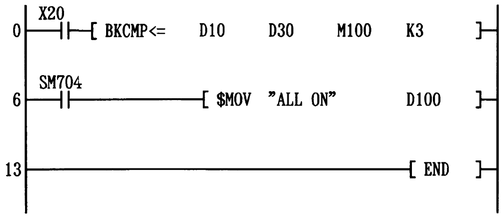
D0 prior to the operation 0 1 0 0 0 0 1 0 0 0 1 0 0 0 0 0

D0 after the operation 0 1 0 0 0 0 1 0 1 0 1 1 0 0 0 0

Bits already in this state do not change [See Function (6)]

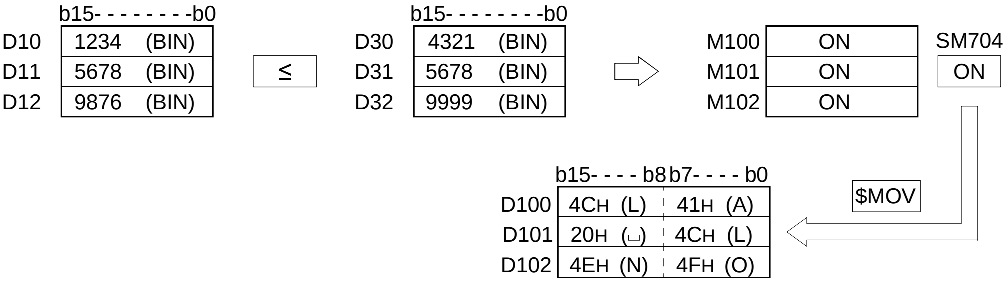
(3) When X20 goes ON, compares the data 3 points from D10 with the data 3 points from D30, and stores the result from M100 onward.
The following program transfers the character string "ALL ON" to D100 onward when all devices from M100 onward have reached the 1 "ON" state.

[Ladder Mode]



[List Mode]

Steps	Instruction	Device
0	LD	X20
1	BKCMP<=	D10 D30 M100 K3
6	LD	SM704
7	$MOV	”ALL ON” D100
13	END	



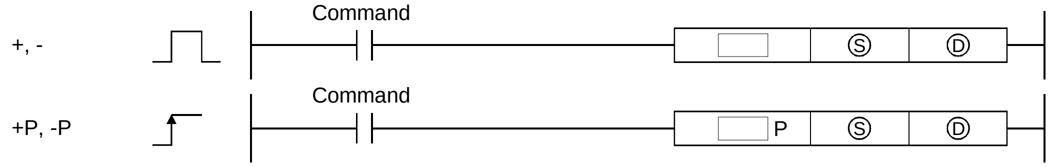
QCPU			QnA	Q4AR
PLC CPU		Process CPU		
Basic	High Performance			
○	○	○	○	○

6.2 Arithmetic Operation Instructions

6.2.1 BIN 16-bit addition and subtraction operations (+, +P, -, -P)

Set Data	Usable Devices									
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant K, H	Other
	Bit	Word		Bit	Word				
Ⓢ	○							○	–
Ⓓ	○							–	–

[Instruction Symbol] [Execution Condition] □ indicates the signs +/-



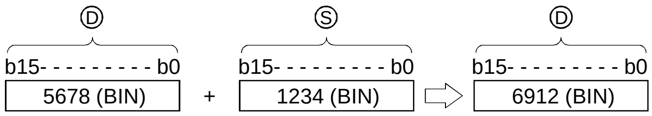
[Set Data]

Set Data	Meaning	Data Type
Ⓢ	Addition or subtraction data, or first number of device storing addition or subtraction data	BIN 16 bits
Ⓓ	Data to be added to or subtracted from, or first number of device storing such data	

[Functions]

+

(1) Adds 16-bit BIN data designated by Ⓓ to 16-bit BIN data designated by Ⓢ and stores the result of the addition at the device designated by Ⓓ.



(2) Values for Ⓢ and Ⓓ can be designated between -32768 and 32767 (BIN, 16 bits).

(3) The judgment of whether data is positive or negative is made by the most significant bit (b15).

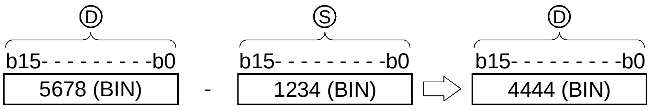
- 0 ....Positive
- 1 ....Negative

(4) The following will happen when an underflow or overflow is generated in an operation result:
The carry flag in this case does not go ON.

- K32767 +K2 → K-32767......A negative value is generated if b15 is 1.
  (H7FFF) (H0002) (H8001)
- K-32768 +K-2 → K32766.......A positive value is generated if b15 is 0.
  (H8000) (HFFFE) (H7FFE)

$\boxed{-}$

(1) Subtracts 16-bit BIN data designated by Ⓓ from 16-bit BIN data designated by Ⓢ and stores the result of the subtraction at the device designated by Ⓓ.



(2) Values for Ⓢ and Ⓓ can be designated between -32768 and 32767 (BIN, 16 bits).

(3) The judgment of whether data is positive or negative is made by the most significant bit (b15).
- 0 .... Positive
- 1 .... Negative

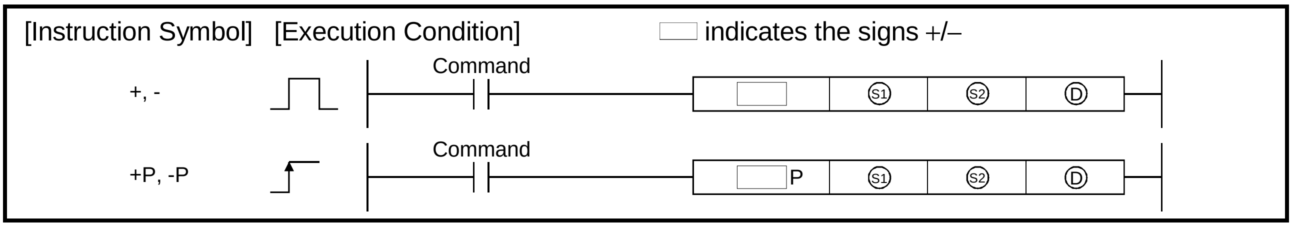
(4) The following will happen when an underflow or overflow is generated in an operation result:
The carry flag in this case does not go ON.

- K-32768 -K2 → K32766.......A positive value is generated if b15 is 0.
  (H8000) (H0002) (H7FFE)
- K32767 -K-2 → K-32767......A negative value is generated if b15 is 1.
  (H7FFF) (H0002) (H8001)

[Operation Errors]

(1) There are no operation errors associated with the +(P) or -(P) instructions.

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant K, H	Other
	Bit	Word		Bit	Word				
Ⓢ1	○							○	–
Ⓢ2	○							○	–
Ⓓ	○							–	–



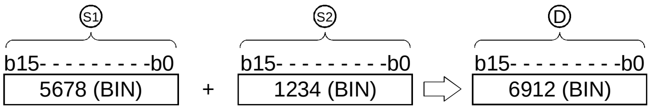
[Set Data]

Set Data	Meaning	Data Type
Ⓢ1	Data to be added to or subtracted from, or the first number of the device storing such data	BIN 16 bits
Ⓢ2	Addition or subtraction data, or first number of device storing addition or subtraction data	
Ⓓ	First number of device storing addition or subtraction data	

[Functions]

+

(1) Adds 16-bit BIN data designated by Ⓢ1 to 16-bit BIN data designated by Ⓢ2 and stores at the device designated by Ⓓ.



(2) Values for Ⓢ1, Ⓢ2 and Ⓓ can be designated from -32768 to 32767 (BIN 16 bits).

(3) The judgment of whether data is positive or negative is made by the most significant bit (b15).

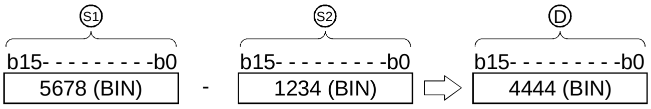
- 0 .... Positive
- 1 .... Negative

(4) The following will happen when an underflow or overflow is generated in an operation result: The carry flag in this case does not go ON.

- K32767 (H7FFF) +K2 (H0002) → K-32767 (H8001)......A negative value is generated if b15 is 1.
- K-32768 (H8000) +K-2 (HFFFE) → K32766 (H7FFE).......A positive value is generated if b15 is 0.

-

(1) Subtracts 16-bit BIN data designated by (S1) from 16-bit BIN data designated by (S2) and stores the result of the subtraction at the device designated by (D).



(2) Values for (S1), (S2) and (D) can be designated from -32768 and 32767 (BIN 16 bits).

(3) The judgment of whether data is positive or negative is made by the most significant bit (b15).

- 0 .... Positive
- 1 .... Negative

(4) The following will happen when an underflow or overflow is generated in an operation result: The carry flag in this case does not go ON.

- K-32768 -K2 → K32766.......A positive value is generated if b15 is 0.
  (H8000) (H0002) (H7FFE)
- K32767 -K-2 → K-32767......A negative value is generated if b15 is 1.
  (H7FFF) (H0002) (H8001)

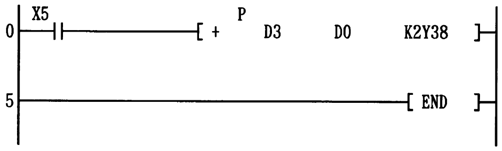
[Operation Errors]

(1) There are no operation errors associated with the +(P) or -(P) instructions.

[Program Example]

(1) The following program adds the contents of D3 and the contents of D0 when X5 goes ON, and outputs result to Y38 to Y3F.

[Ladder Mode]

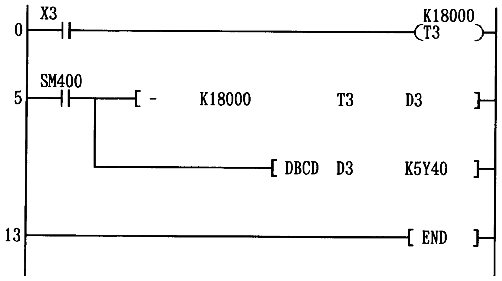


[List Mode]

Steps	Instruction	Device
0	LD	X5
1	+P	D3 D0 K2Y38
5	END	

(2) The following program outputs the difference between the set value for timer T3 and its present value to Y40 to Y53.

[Ladder Mode]



[List Mode]

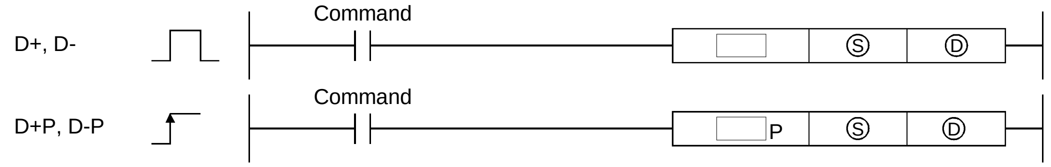
Steps	Instruction	Device
0	LD	X3
1	OUT	T3 K18000
5	LD	SM400
6	-	K18000 T3 D3
10	DBCD	D3 K5Y40
13	END	

QCPU			QnA	Q4AR
PLC CPU		Process CPU		
Basic	High Performance			
○	○	○	○	○

6.2.2 BIN 32-bit addition and subtraction operations (D+, D+P, D-, D-P)

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant K, H	Other
	Bit	Word		Bit	Word				
Ⓢ	○							○	–
Ⓓ	○							–	–

[Instruction Symbol] [Execution Condition] □ indicates the signs D+/D-



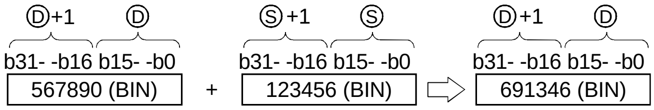
[Set Data]

Set Data	Meaning	Data Type
Ⓢ	Addition or subtraction data, or first number of device storing addition or subtraction data	BIN 32 bits
Ⓓ	Data to be added to or subtracted from, or first number of device storing such data	

[Functions]

D+

(1) Adds 32-bit BIN data designated by Ⓓ to 32-bit BIN data designated by Ⓢ, and stores the result of the addition at the device designated by Ⓓ.



(2) The values for Ⓢ and Ⓓ can be designated at between -2147483648 and 2147483647 (BIN 32 bits).

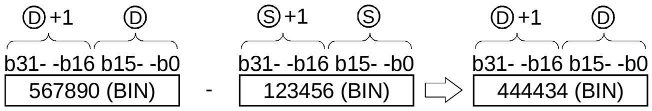
(3) Judgment of whether the data is positive or negative is made on the basis of the most significant bit (b31).
- 0 ....Positive
- 1 ....Negative

(4) The following will happen when an underflow or overflow is generated in an operation result: The carry flag in this case does not go ON.

- K2147483647 (H7FFFFFFF) +K2 (H2) → K-2147483647 (H80000001) ....Because b31 is 1, the value is negative.
- K-2147483648 (H80000000) +K-2 (HFFFE) → K2147483646 (H7FFFFFFE) ....Because b31 is 0, the value is positive.

D-

(1) Subtracts 32-bit BIN data designated by Ⓓ from 32-bit BIN data designated by Ⓢ and stores the result of the subtraction at the device designated by Ⓓ.



(2) The values for Ⓢ and Ⓓ can be designated at between -2147483648 and 2147483647 (BIN 32 bits).

(3) Judgment of whether the data is positive or negative is made on the basis of the most significant bit (b31).

- 0 .... Positive
- 1 .... Negative

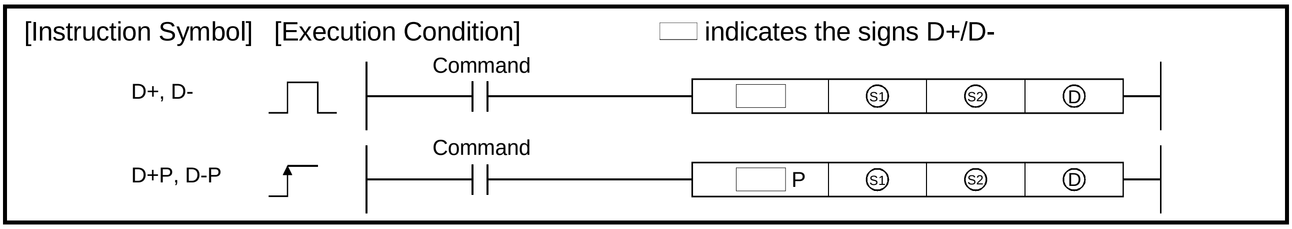
(4) The following will happen when an underflow or overflow is generated in an operation result: The carry flag in this case does not go ON.

- K-2147483648 (H80000000) -K2 (H2) → K2147483646 (H7FFFFFFE) ....Because b31 is 0, the value is positive.
- K2147483647 (H7FFFFFFF) -K-2 (HFFFE) → K-2147483647 (H80000001) ....Because b31 is 1, the value is negative.

[Operation Errors]

(1) There are no operation errors associated with the +(P) or -(P) instructions.

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant K, H	Other
	Bit	Word		Bit	Word				
Ⓢ1	○							○	–
Ⓢ2	○							○	–
Ⓓ	○							–	–



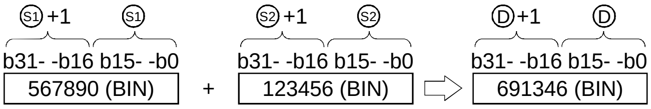
[Set Data]

Set Data	Meaning	Data Type
Ⓢ1	Data to be added to or subtracted from, or the first number of the device storing such data	BIN 32 bits
Ⓢ2	Addition or subtraction data, or first number of device storing addition or subtraction data	
Ⓓ	First number of device storing addition or subtraction data	

[Functions]

D+

(1) Adds 32-bit BIN data designated by Ⓢ1 to 32-bit BIN data designated by Ⓢ2 and stores at the device designated by Ⓓ.



(2) The values for Ⓢ1, Ⓢ2, and Ⓓ can be designated at between -2147483648 and 2147483647 (BIN 32 bits).

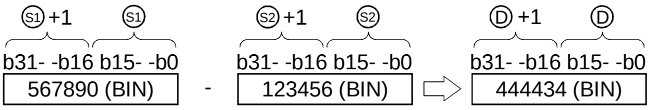
(3) Judgment of whether the data is positive or negative is made on the basis of the most significant bit (b31).

(4) The following will happen when an underflow or overflow is generated in an operation result: The carry flag in this case does not go ON.

```
• K2147483647  +K2 ——→ K-2147483647.......Because b31 is 0,
  (H7FFFFFFF)  (H2)      (H80000001)        the value is positive.
• K-2147483648 +K-2 ——→ K2147483646........Because b31 is 1,
  (H80000000)  (HFFFE)   (HFFFE)            the value is negative.
```

D-

(1) Subtracts 32-bit BIN data designated by Ⓢ1 from 32-bit BIN data designated by Ⓢ2 and stores the result of the subtraction at the device designated by Ⓓ.



(2) The values for Ⓢ1, Ⓢ2, and Ⓓ can be designated at between -2147483648 and 2147483647 (BIN 32 bits).

(3) Judgment of whether the data is positive or negative is made on the basis of the most significant bit (b31).

• 0 .... Positive

• 1 .... Negative

(4) The following will happen when an underflow or overflow is generated in an operation result: The carry flag in this case does not go ON.

• K-2147483648 (H80000000) -K2 (H2) → K2147483646 (H7FFFFFFE)........Because b31 is 0, the value is positive.

• K2147483647 (H7FFFFFFF) -K-2 (HFFFE) → K-2147483647 (HFFFE).......Because b31 is 1, the value is negateve.

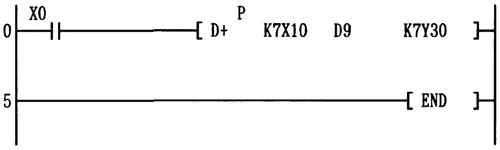
[Operation Errors]

(1) There are no operation errors associated with the +(P) or -(P) instructions.

[Program Example]

(1) The following program adds 28-bit data from X10 to X2B to the data at D9 and D10 when X0 goes ON, and outputs the result of the operation to Y30 to Y4B.

[Ladder Mode]

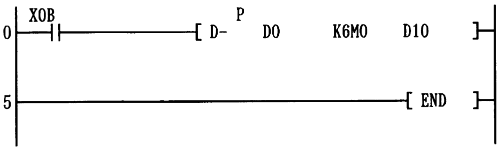


[List Mode]

Steps	Instruction	Device
0	LD	X0
1	D+P	K7X10
		D9
		K7Y30
5	END	

(2) The following program subtracts the data from M0 to M23 from the data at D0 and D1 when XB goes ON, and stores the result at D10 and D11.

[Ladder Mode]



[List Mode]

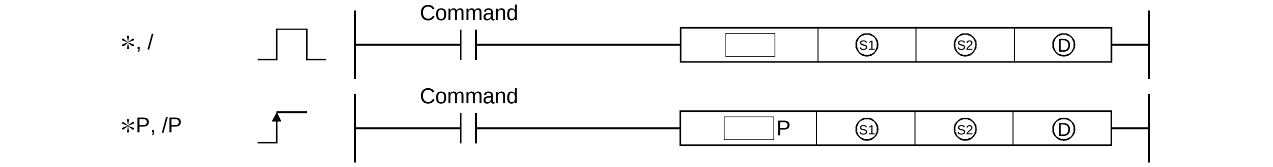
Steps	Instruction	Device
0	LD	X0B
1	D-P	D0
		K6M0
		D10
5	END	

QCPU			QnA	Q4AR
PLC CPU		Process CPU		
Basic	High Performance			
○	○	○	○	○

6.2.3 BIN 16-bit multiplication and division operations (∗ ,∗P, /, /P)

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant K, H	Other
	Bit	Word		Bit	Word				
Ⓢ1	○							○	–
Ⓢ2	○							○	–
Ⓓ	○							–	–

[Instruction Symbol]   [Execution Condition]   □ indicates the signs ∗ or /



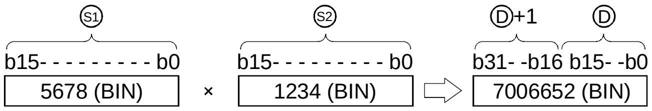
[Set Data]

Set Data	Meaning	Data Type
Ⓢ1	Data that will be multiplied or divided, or the first number of the device storing data that will be multiplied or divided	BIN 16 bits
Ⓢ2	Data to multiply or divide by, or the first number of device storing such data	
Ⓓ	First number of the device storing the operation results of multiplication or division operation	BIN 32 bits

[Functions]

∗

(1) Multiplies BIN 16-bit data designated by Ⓢ1 and BIN 16-bit data designated by Ⓢ2, and stores the result in the device designated by Ⓓ.



(2) If Ⓓ is a bit device, designation is made from the lower bits.

Example K1...........Lower 4 bits (b0 to 3)
K4...........Lower 16 bits (b0 to 15)
K8...........Lower 32 bits (b0 to 31)

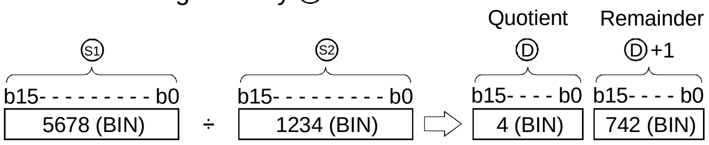
(3) The values for Ⓢ1, Ⓢ2, and Ⓓ can be designated at between -32768 and 32767 (BIN 16 bits).

(4) Judgments whether Ⓢ1, Ⓢ2, and (D) are positive or negative are made on the basis of the most significant bit (b15 for Ⓢ1, and Ⓢ2, for Ⓓ and b31).

- 0 .... Positive
- 1 .... Negative

/

(1) Divides BIN 16-bit data designated by (S1) and BIN 16-bit data designated by (S2), and stores the result in the device designated by (D).



(2) If a word device has been used, the result of the division operation is stored as 32 bits, and both the quotient and remainder are stored; if a bit device has been used, 16 bits are used and only the quotient is stored.

Quotient .... Stored at the lower 16 bits
Remainder...... Stored at the higher 16 bits
(Can be stored only when a word device has been used)

(3) The values for (S1), (S2), and (D) can be designated at between -32768 and 32767 (BIN 16 bits).

(4) Judgment whether values for (S1), (S2), and (D) are positive or negative is made on the basis of the most significant bit (b15 for (S1) and (S2), and b15 for (D)).

- 0 .... Positive
- 1 .... Negative

[Operation Errors]

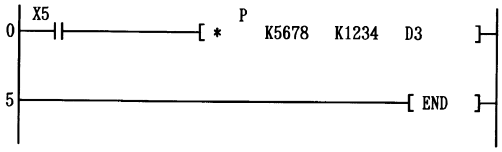
(1) In the following cases an operation error occurs, the error flag (SM0) turns ON, and an error code is stored at SD0.

- Attempt to divide (S2) by 0. (Error code: 4100)

[Program Example]

(1) The following program divides "5678" by "1234" when X5 goes ON, and stores the result at D3 and D4.

[Ladder Mode]

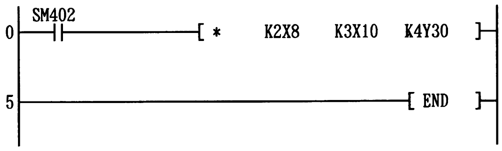


[List Mode]

Steps	Instruction	Device
0	LD	X5
1	*P	K5678 K1234 D3
5	END	

(2) The following program divides BIN data at X8 to XF by BIN data at X10 to X1B, and outputs the result of the division operation to Y30 to Y3F.

[Ladder Mode]

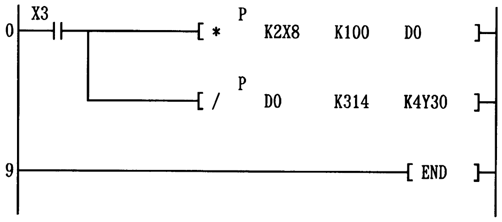


[List Mode]

Steps	Instruction	Device
0	LD	SM402
1	*	K2X8 K3X10 K4Y30
5	END	

(3) The following program outputs the value resulting when the data at X8 to XF is divided by 3.14 to Y30 to Y3F when X3 is ON.

[Ladder Mode]



[List Mode]

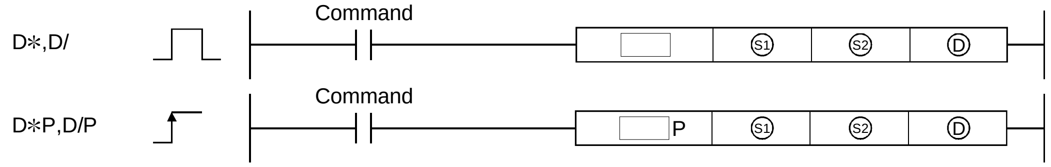
Steps	Instruction	Device
0	LD	X3
1	*P	K2X8 K100 D0
5	/P	D0 K314 K4Y30
9	END	

QCPU				QnA	Q4AR
PLC CPU			Process CPU		
Basic	High Performance				
○	○		○	○	○

6.2.4 BIN 32-bit multiplication and division operations (D∗, D∗P, D/, D/P)

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant K, H	Other
	Bit	Word		Bit	Word				
Ⓢ1	○			○					–
Ⓢ2	○			○					–
Ⓓ	○			–					–

[Instruction Symbol] [Execution Condition] □ indicates the signs D∗ or D/



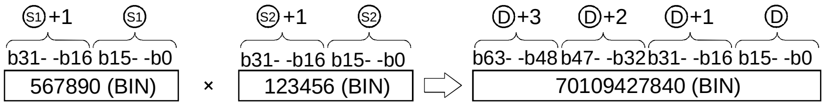
[Set Data]

Set Data	Meaning	Data Type
Ⓢ1	Data that will be multiplied or divided, or the head number of the device storing data that will be multiplied or divided	BIN 32 bits
Ⓢ2	Data to multiply or divide by, or the head number of device storing such data	
Ⓓ	Head number of the device storing the operation results of multiplication or division operation	BIN 64 bits

[Functions]

D∗

(1) Multiplies BIN 32-bit data designated by Ⓢ1 and BIN 32-bit data designated by Ⓢ2, and stores the result in the device designated by Ⓓ.



(2) If Ⓓ is a bit device, only the lower 32 bits of the multiplication result will be considered, and the upper 32 bits cannot be designated.

Example K1...........Lower 4 bits (b0 to 3)
K4...........Lower 16 bits (b0 to 15)
K8...........Lower 32 bits (b0 to 31)

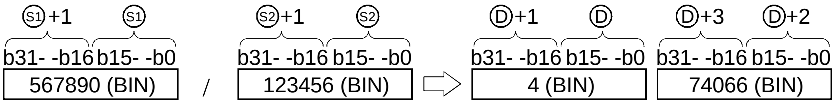
If the upper 32 bits of the bit device are required for the result of the multiplication operation, first temporarily store the data in a word device, then transfer the word device data to the bit device by designating (Ⓓ+2) and (Ⓓ+3) data.

(3) The values for Ⓢ1, Ⓢ2, and Ⓓ can be designated at between -2147483648 to 2147483647 (BIN 32 bits).

(4) Judgment whether values for Ⓢ1, Ⓢ2, and Ⓓ are positive or negative are made on the basis of the most significant bit (b31 for Ⓢ1 and Ⓢ2, and b63 for Ⓓ).
- 0 .... Positive
- 1 .... Negative

D/

(1) Divides BIN 32-bit data designated by Ⓢ1 and BIN 32-bit data designated by Ⓢ2, and stores the result in the device designated by Ⓓ.



(2) If a word device has been used, the result of the division operation is stored as 64 bits, and both the quotient and remainder are stored; if a bit device has been used, 32 bits are used and only the quotient is stored.

Quotient .... Stored at the lower 32 bits
Remainder ...... Stored at the higher 32 bits
(Can be stored only when a word device has been used)

(3) The values for Ⓢ1 and Ⓢ2 can be designated at between -2147483648 to 2147483647 (BIN 32 bits).

(4) Judgment whether values for Ⓢ1, Ⓢ2, Ⓓ, and Ⓓ+2 are positive or negative is made on the basis of the most significant bit (b31).
(A sign is used with both the quotient and the remainder)
- 0 .... Positive
- 1 .... Negative

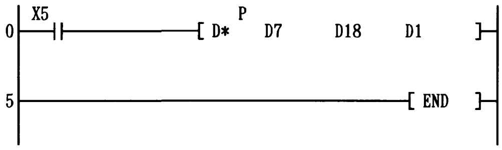
[Operation Errors]

(1) In the following cases an operation error occurs, the error flag (SM0) turns ON, and an error code is stored at SD0.
- Attempt to divide Ⓢ2 by 0. (Error code: 4100)

[Program Example]

(1) The following program divides the BIN data at D7 and D8 by the BIN data at D18 and D19 when X5 is ON, and stores the result at D1 to D4.

[Ladder Mode]

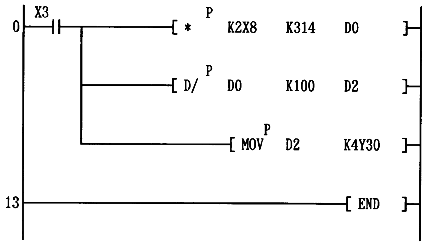


[List Mode]

Steps	Instruction	Device
0	LD	X5
1	D*P	D7 D18 D1
5	END	

(2) The following program outputs the value resulting when the data at X8 to XF is multiplied by 3.14 to Y30 to Y3F when X3 is ON.

[Ladder Mode]



[List Mode]

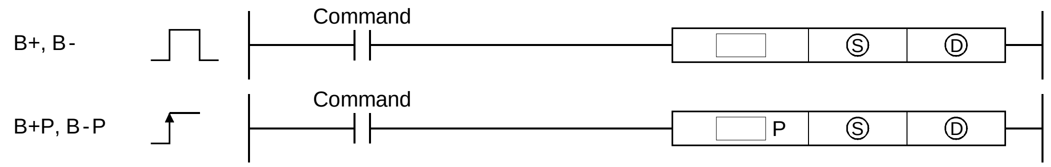
Steps	Instruction	Device
0	LD	X3
1	*P	K2X8 K314 D0
5	D/P	D0 K100 D2
10	MOVP	D2 K4Y30
13	END	

QCPU			QnA	Q4AR
PLC CPU		Process CPU		
Basic	High Performance			
○	○	○	○	○

6.2.5 BCD 4-digit addition and subtraction operations (B+, B+P, B-, B-P)

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant K, H	Other
	Bit	Word		Bit	Word				
Ⓢ	○							○	–
Ⓓ	○							–	–

[Instruction Symbol] [Execution Condition] □ indicates the signs B+ or B-



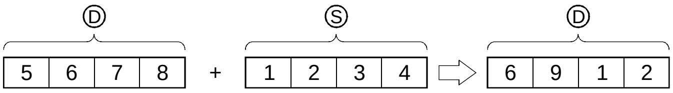
[Set Data]

Set Data	Meaning	Data Type
Ⓢ	Addition or subtraction data, or head number of device storing addition or subtraction data	BCD 4-digit
Ⓓ	Data to be added to or subtracted from, or head number of device storing such data	

[Functions]

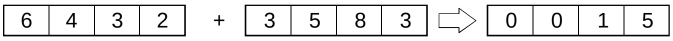
B+

(1) Adds the BCD 4-digit data designated by Ⓓ and the BCD 4-digit data designated by Ⓢ, and stores the result of the addition at the device designated by Ⓓ.



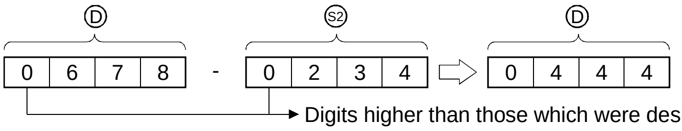
(2) The values for Ⓢ and Ⓓ can be between 0 to 9999 (BCD 4-digit).

(3) If the result of the addition operation exceeds 9999, the higher bits are ignored. The carry flag in this case does not go ON.



B-

(1) Subtracts the BCD 4-digit data designated by Ⓓ and the BCD 4-digit data designated by Ⓢ, and stores the result of the subtraction at the device designated by Ⓓ.



Digits higher than those which were designated will be read as 0.

(2) The values for Ⓢ and Ⓓ can be between 0 to 9999 (BCD 4-digit).

(3) The following will result if an underflow is generated by the subtraction operation:
The carry flag in this case does not go ON.

0	0	0	1	-	0	0	0	3	⇨	9	9	9	8

[Operation Errors]

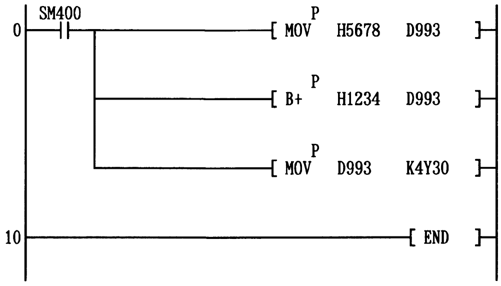
(1) In the following cases an operation error occurs, the error flag (SM0) turns ON, and an error code is stored at SD0.

- The Ⓢ or Ⓓ BCD data is outside the 0 to 9999 range. (Error code: 4100)

[Program Example]

(1) The following program adds BCD data 5678 and 1234, stores it at D993, and at the same time outputs it to from Y30 to Y3F.

[Ladder Mode]



Stores 5678 at D993

Adds BCD 1234 and D993, and stores the result in D993

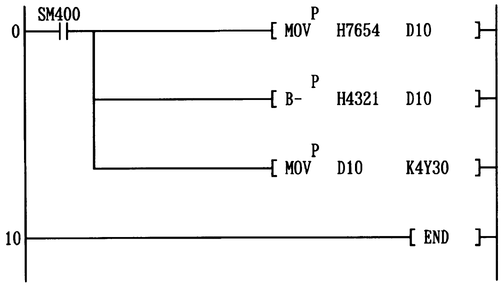
Outputs data at D993 to Y30 to Y3F

[List Mode]

Steps	Instruction	Device
0	LD	SM400
1	MOVP	H5678 D993
4	B+P	H1234 D993
7	MOVP	D993 K4Y30
10	END	

(2) The following program subtracts the BCD data 4321 from 7654, stores the result at D10, and at the same time outputs it to Y30 to Y3F.

[Ladder Mode]



Stores 7654 at D10 as BCD

Subtracts data in D10 from BCD 4321 and stores result at D10

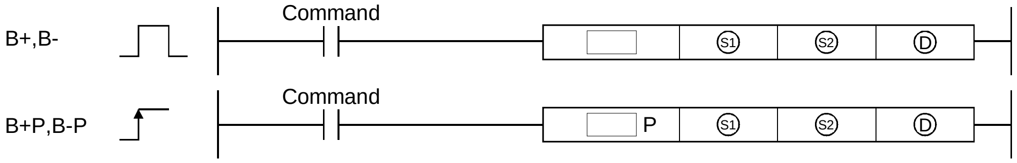
Outputs data at D10 to Y30 to Y3F

[List Mode]

Steps	Instruction	Device
0	LD	SM400
1	MOVP	H7654 D10
4	B-P	H4321 D10
7	MOVP	D10 K4Y30
10	END	

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant K, H	Other
	Bit	Word		Bit	Word				
Ⓢ1	○							○	—
Ⓢ2	○							○	—
Ⓓ	○							—	—

[Instruction Symbol] [Execution Condition] □ indicates the signs B+ or B-



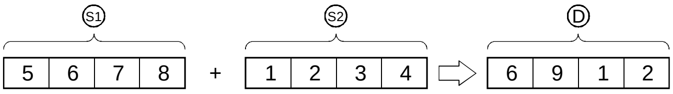
[Set Data]

Set Data	Meaning	Data Type
Ⓢ1	Data to be added to or subtracted from, or the head number of the device storing such data	BCD 4-digit
Ⓢ2	Addition or subtraction data, or head number of device storing addition or subtraction data	
Ⓓ	Head number of device storing addition or subtraction data	

[Functions]

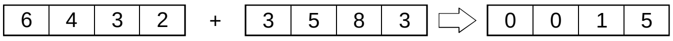
B+

(1) Adds the BCD 4-digit data designated by Ⓢ1 and the BCD 4-digit data designated by Ⓢ2, and stores the result of the addition at the device designated by Ⓓ.



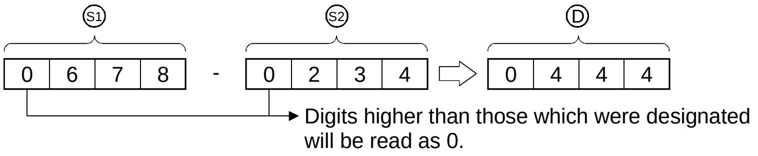
(2) The values for Ⓢ1, Ⓢ2, and Ⓓ can be between 0 to 9999 (BCD 4-digit data).

(3) If the result of the addition operation exceeds 9999, the higher bits are ignored. The carry flag in this case does not go ON.



B-

(1) Subtracts the BCD 4-digit data designated by Ⓢ1 and the BCD 4-digit data designated by Ⓢ2, and stores the result of the subtraction at the device designated by Ⓓ.



Digits higher than those which were designated will be read as 0.

(2) The values for Ⓢ1, Ⓢ2, and Ⓓ can be between 0 to 9999 (BCD 4-digit data).

(3) The following will result if an underflow is generated by the subtraction operation:
The carry flag in this case does not go ON.

0	0	0	1	-	0	0	0	3	⇨	9	9	9	8

[Operation Errors]

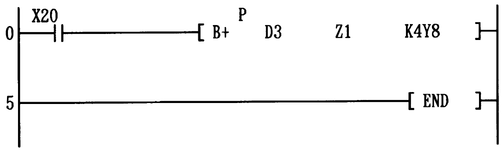
(1) In the following cases an operation error occurs, the error flag (SM0) turns ON, and an error code is stored at SD0.

- The Ⓢ1, Ⓢ2, or Ⓓ BCD data is outside the 0 to 9999 range. (Error code: 4100)

[Program Example]

(1) The following program adds the D3 BCD data and the Z1 BCD data when X20 goes ON, and outputs the result to Y8 to Y17.

[Ladder Mode]

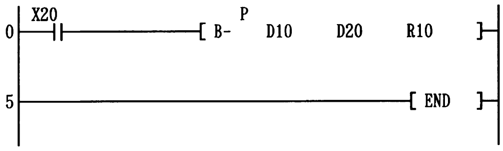


[List Mode]

Steps	Instruction	Device
0	LD	X20
1	B+P	D3 Z1 K4Y8
5	END	

(2) The following program subtracts the BCD data at D20 from the BCD data at D10 when X20 goes ON, and stores the result at R10.

[Ladder Mode]



[List Mode]

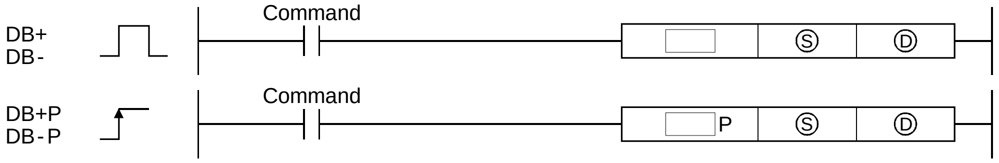
Steps	Instruction	Device
0	LD	X20
1	B-P	D10 D20 R10
5	END	

QCPU			QnA	Q4AR
PLC CPU		Process CPU		
Basic	High Performance			
○	○	○	○	○

6.2.6 BCD 8-digit addition and subtraction operations (DB+, DB+P, DB-, DB-P)

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant K, H	Other
	Bit	Word		Bit	Word				
Ⓢ	○							○	–
Ⓓ	○							–	–

[Instruction Symbol]   [Execution Condition]   □ indicates the signs DB+ or DB-



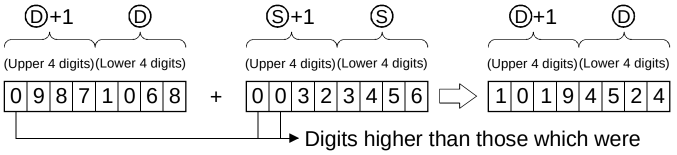
[Set Data]

Set Data	Meaning	Data Type
Ⓢ	Addition or subtraction data, or head number of device storing addition or subtraction data	BCD 8-digit
Ⓓ	Data to be added to or subtracted from, or head number of device storing such data	

[Functions]

DB+

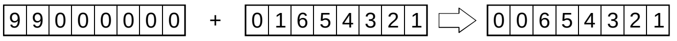
(1) Adds the BCD 8-digit data designated by Ⓓ and the BCD 8-digit data designated by Ⓢ, and stores the result of the addition at the device designated by Ⓓ.


Ⓓ+1
Ⓓ
(Upper 4 digits)(Lower 4 digits)
Ⓢ+1
Ⓢ
(Upper 4 digits)(Lower 4 digits)
Ⓓ+1
Ⓓ
(Upper 4 digits)(Lower 4 digits)
0 9 8 7 1 0 6 8
+
0 0 3 2 3 4 5 6
1 0 1 9 4 5 2 4

Digits higher than those which were designated will be read as 0.

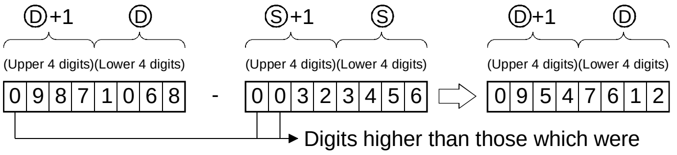
(2) The values for Ⓢ and Ⓓ can be between 0 to 99999999 (BCD 8-digit data).

(3) If the result of the addition operation exceeds 99999999, the upper bits will be ignored.
The carry flag in this case does not go ON.



DB-

(1) Subtracts the BCD 8-digit data designated by Ⓓ and the BCD 8-digit data designated by Ⓢ, and stores the result of the subtraction at the device designated by Ⓓ.


Ⓓ+1
Ⓓ
(Upper 4 digits)(Lower 4 digits)
Ⓢ+1
Ⓢ
(Upper 4 digits)(Lower 4 digits)
Ⓓ+1
Ⓓ
(Upper 4 digits)(Lower 4 digits)
0 9 8 7 1 0 6 8
-
0 0 3 2 3 4 5 6
0 9 5 4 7 6 1 2

Digits higher than those which were designated will be read as 0.

(2) The values for Ⓢ and Ⓓ can be between 0 to 99999999 (BCD 8-digit).

(3) The following will result if an underflow is generated by the subtraction operation:
The carry flag in this case does not go ON.

1	2	3	4	5	6	7	8	-	1	2	3	4	5	6	7	9	⇨	9	9	9	9	9	9	9	9

[Operation Errors]

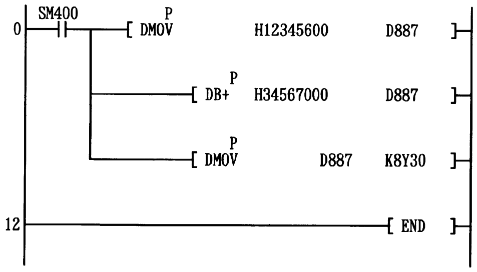
(1) In the following cases an operation error occurs, the error flag (SM0) turns ON, and an error code is stored at SD0.

- The Ⓢ or Ⓓ BCD data is outside the 0 to 99999999 range. (Error code: 4100)

[Program Example]

(1) The following program adds the BCD data 12345600 and 34567000, stores the result at D887 and D888, and at the same time outputs them to from Y30 to Y4F.

[Ladder Mode]



Stores 12345600 at D887,D888 as BCD

Adds BCD 34567000 and D887,D888
and stores the result in D887,D888

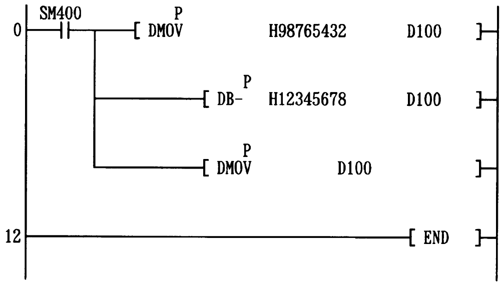
Outputs data at D887,D888 to Y30 to Y4F

[List Mode]

Steps	Instruction	Device
0	LD	SM400
1	DMOVP	H12345600 D887
5	DB+P	H34567000 D887
9	DMOVP	D887 K8Y30
12	END	

(2)The following program subtracts the BCD data 98765432 from 12345678, stores the result at D100 and D101, and at the same time outputs it from Y30 to Y4F.

[Ladder Mode]



Stores 98765432 at D100,D101 as BCD data.

Subtracts 12345678 from data in D100,D101
and stores result as BCD data at D100,D101

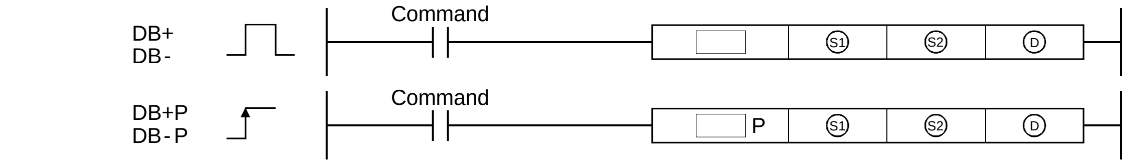
Outputs data at D100,D101 to Y30 to Y4F

[Ladder Mode]

Steps	Instruction	Device
0	LD	SM400
1	DMOVP	H98765432 D100
5	DB-P	H12345678 D100
9	DMOVP	D100 K8Y30
12	END	

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant K, H	Other
	Bit	Word		Bit	Word				
Ⓢ1	○							○	—
Ⓢ2	○							○	—
Ⓓ	○							—	—

[Instruction Symbol] [Execution Condition] ▭ indicates the signs DB+ or DB-



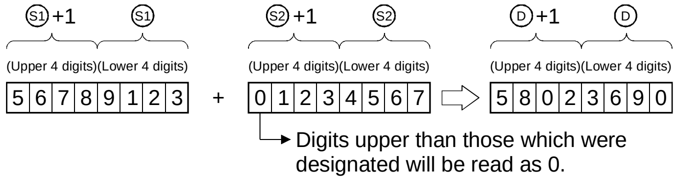
[Set Data]

Set Data	Meaning	Data Type
Ⓢ1	Data to be added to or subtracted from, or the head number of the device storing such data	BCD 8-digit
Ⓢ2	Addition or subtraction data, or head number of device storing addition or subtraction data	
Ⓓ	Head number of device storing addition or subtraction data	

[Functions]

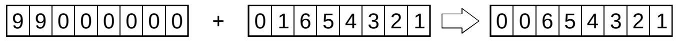
DB+

(1) Adds the BCD 8-digit data designated by Ⓢ1 and the BCD 8-digit data designated by Ⓢ2, and stores the result of the addition at the device designated by Ⓓ.


Ⓢ1+1
Ⓢ1
(Upper 4 digits)(Lower 4 digits)
5 6 7 8 9 1 2 3
+
Ⓢ2+1
Ⓢ2
(Upper 4 digits)(Lower 4 digits)
0 1 2 3 4 5 6 7
Ⓓ+1
Ⓓ
(Upper 4 digits)(Lower 4 digits)
5 8 0 2 3 6 9 0
Digits upper than those which were designated will be read as 0.

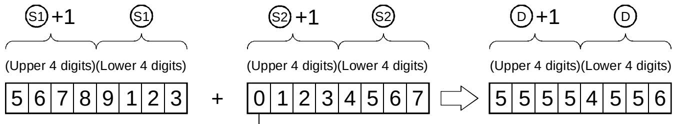
(2) The values for Ⓢ1, Ⓢ2, and Ⓓ can be between 0 to 99999999 (BCD 8 digits).

(3) If the result of the addition operation exceeds 99999999, the upper bits will be ignored. The carry flag in this case does not go ON.



DB-

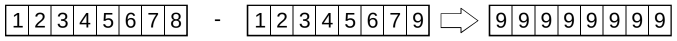
(1) Subtracts the BCD 8-digit data designated by Ⓢ1 and the BCD 8-digit data designated by Ⓢ2, and stores the result of the subtraction at the device designated by Ⓓ.


Ⓢ1+1
Ⓢ1
(Upper 4 digits)(Lower 4 digits)
5 6 7 8 9 1 2 3
+
Ⓢ2+1
Ⓢ2
(Upper 4 digits)(Lower 4 digits)
0 1 2 3 4 5 6 7
Ⓓ+1
Ⓓ
(Upper 4 digits)(Lower 4 digits)
5 5 5 5 4 5 5 6

Digits higher than those which were designated will be read as 0.

(2) The values for Ⓢ1, Ⓢ2, and Ⓓ can be between 0 to 99999999 (BCD 8 digits).

(3) The following will result if an underflow is generated by the subtraction operation: The carry flag in this case does not go ON.



[Operation Errors]

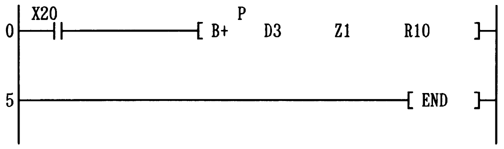
(1) In the following cases an operation error occurs, the error flag (SM0) turns ON, and an error code is stored at SD0.

- The Ⓢ1, Ⓢ2, or Ⓓ BCD data is outside the 0 to 99999999 range. (Error code: 4100)

[Program Example]

(1) The following program adds the BCD data at D3 and D4 to the BCD data at Z1 and Z2 when X20 goes ON, and stores the result at R10 and R11.

[Ladder Mode]



[List Mode]

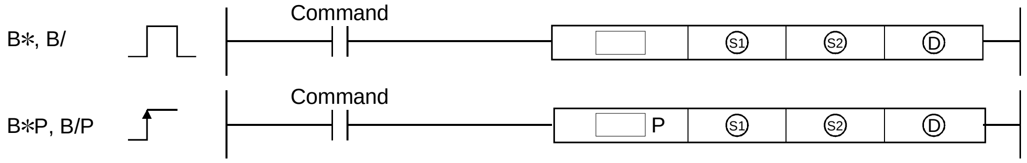
Steps	Instruction	Device
0	LD	X20
1	B+P	D3 Z1 R10
5	END	

QCPU					
PLC CPU		Process CPU	QnA	Q4AR	
Basic	High Performance				
○	○	○	○	○	

6.2.7 BCD 4-digit multiplication and division operations (B∗, B∗P, B/, B/P)

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant K, H	Other
	Bit	Word		Bit	Word				
Ⓢ1	○							○	–
Ⓢ2	○							○	–
Ⓓ	○							–	–

[Instruction Symbol] [Execution Condition] □ indicates the signs B∗ or B/



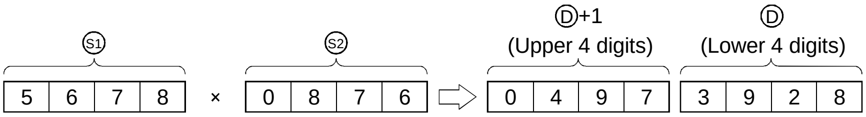
[Set Data]

Set Data	Meaning	Data Type
Ⓢ1	Data that will be multiplied or divided, or the head number of the device storing data that will be multiplied or divided	BCD 4-digit
Ⓢ2	Data to multiply or divide by, or the head number of device storing such data	
Ⓓ	Head number of the device storing the operation results of multiplication or division operation	BCD 8-digit

[Functions]

B∗

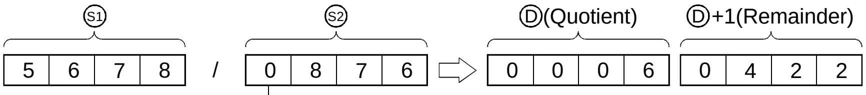
(1) Multiplies BCD data designated by Ⓢ1 and BCD data designated by Ⓢ2, and stores the result in the device designated by Ⓓ.



(2) Values for Ⓢ1 and Ⓢ2 can be set from 0 to 9999 (BCD 4 digits).

B/

(1) Divides BCD data designated by Ⓢ1 and BCD data designated by Ⓢ2, and stores the result in the device designated by Ⓓ.



Digits higher than those which were designated will be read as 0.

(2) Uses 32 bits to store the result of the division as quotient and remainder
Quotient (BCD 4 digits) .... Stored at the lower 16 bits
Remainder (BCD 4 digits) .... Stored at the upper 16 bits

(3) If Ⓓ has been designated as a bit device, the remainder of the operation will not be stored.

[Operation Errors]

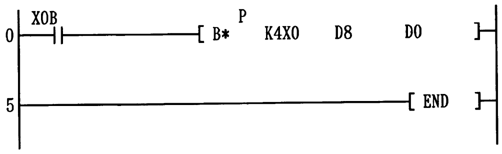
(1) In the following cases an operation error occurs, the error flag (SM0) turns ON, and an error code is stored at SD0.

- The S1 or S2 data is outside the 0 to 9999 range. (Error code: 4100)
- Attempt to divide S2 by 0. (Error code: 4100)

[Program Example]

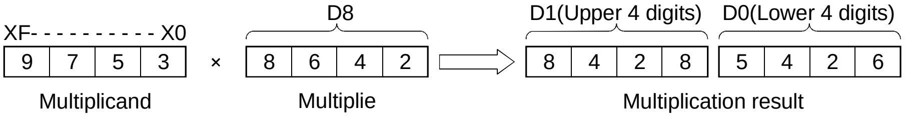
(1) The following program multiplies the BCD data at X0 to XF and the BCD data at D8 when X0B goes ON, and stores the result at D0 and D1.

[Ladder Mode]



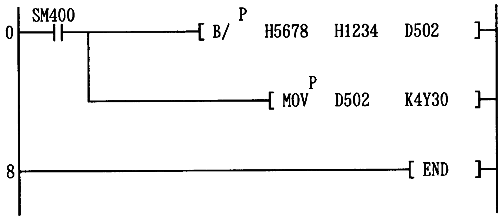
[List Mode]

Steps	Instruction	Device
0	LD	X0B
1	B*P	K4X0 D8 D0
5	END	



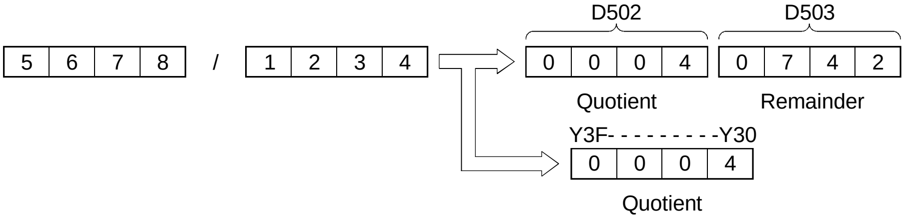
(2) The following program divides 5678 by the BCD data 1234, stores the result at D502 and D503, and at the same time outputs the quotient to Y30 to Y3F.

[Ladder Mode]



[List Mode]

Steps	Instruction	Device
0	LD	SM400
1	B/P	H5678 H1234 D502
5	MOVP	D502 K4Y30
8	END	

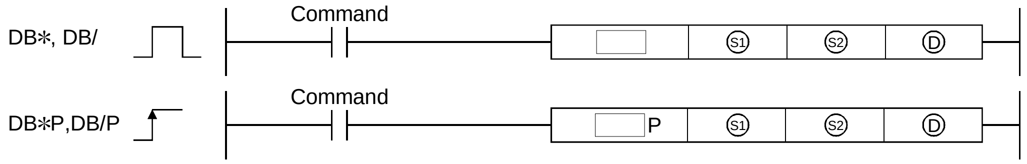


QCPU			QnA	Q4AR
PLC CPU		Process CPU		
Basic	High Performance			
○	○	○	○	○

6.2.8 BCD 8-digit multiplication and division operations (DB∗, DB∗P, DB/, DB/P)

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant K, H	Other
	Bit	Word		Bit	Word				
Ⓢ1	○			○					–
Ⓢ2	○			○					–
Ⓓ	○			–					–

[Instruction Symbol] [Execution Condition] □ indicates the signs DB∗ or DB/



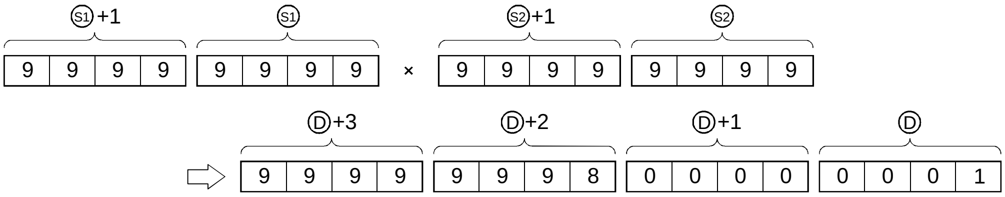
[Set Data]

Set Data	Meaning	Data Type
Ⓢ1	Data that will be multiplied or divided, or the head number of the device storing data that will be multiplied or divided	BCD 8 digits
Ⓢ2	Data to multiply or divide by, or the head number of device storing such data	
Ⓓ	Head number of the device storing the operation results of multiplication or division operation	BCD 16 digits

[Functions]

DB∗

(1) Multiplies the BCD 8-digit data designated by Ⓢ1 and the BCD 8-digit data designated by Ⓢ2, and stores the product at the device designated by Ⓓ.



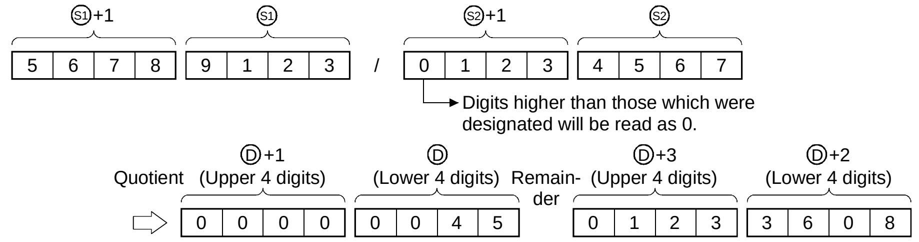
(2) If Ⓓ has designated a bit device, the lower 8 digits (lower 32 bits) will be used for the product, and the higher 8 digits (upper 32 bits) cannot be designated.

K1 · · · Lower 1 digit (b0 to 3)
K4 · · · Lower 4 digits (b0 to 15)
K8 · · · Lower 8 digits (b0 to 31)

(3) The values for Ⓢ1 and Ⓢ2 can be designated from 0 to 99999999 (8 digit BCD).

DB/

(1) Divides 8-digit BCD data designated by Ⓢ1 and 8-digit BCD data designated by Ⓢ2, and stores the result in the device designated by Ⓓ.



(2) 64 bits are used for the result of the division operation, and stored as quotient and remainder.
Quotient (BCD 8 digits) .... Stored at the lower 32 bits
Remainder (BCD 8 digits) .... Stored at the upper 32 bits

(3) If Ⓓ has been designated as a bit device, the remainder of the operation will not be stored.

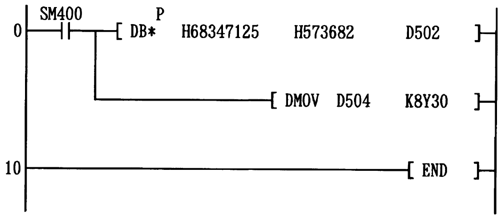
[Operation Errors]

(1) In the following cases an operation error occurs, the error flag (SM0) turns ON, and an error code is stored at SD0.
- The Ⓢ1 or Ⓢ2 data is outside the 0 to 99999999 range. (Error code: 4100)
- Attempt to divide Ⓢ2 by 0. (Error code: 4100)

[Program Example]

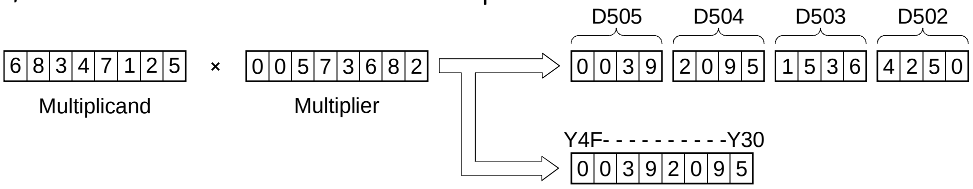
(1) The following program multiplies the BCD data 67347125 and 573682, stores the result from D502 to D505, and at the same time outputs the upper 8 digits to Y30 to Y4F.

[Ladder Mode]



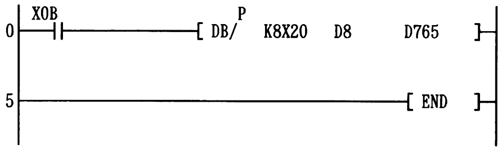
[List Mode]

Steps	Instruction	Device
0	LD	SM400
1	DB∗P	H68347125 H573682 D502
7	DMOV	D504 K8Y30
10	END	



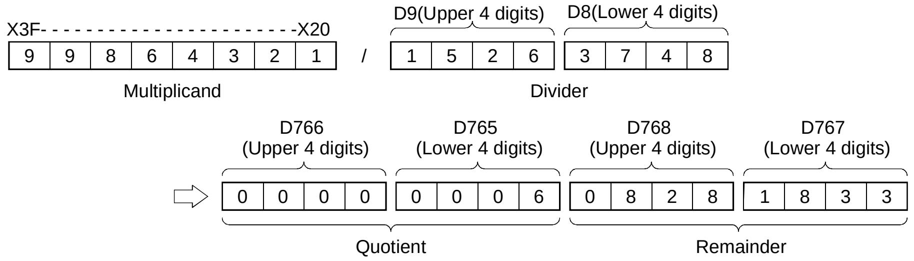
(2) The following program divides the BCD data from X20 to 3F by the BCD data at D8 and D9 when X0B goes ON, and stores the result from D765 to D768.

[Ladder Mode]



[List Mode]

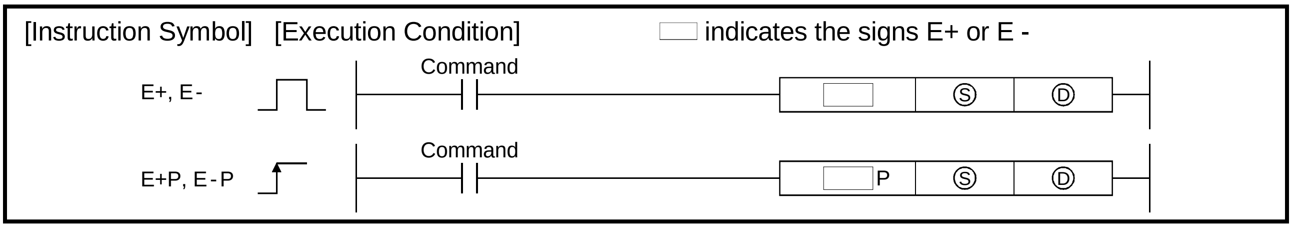
Steps	Instruction	Device
0	LD	X0B
1	DB/P	K8X20 D8 D765
5	END	



QCPU			QnA	Q4AR
PLC CPU		Process CPU		
Basic	High Performance			
×	○	○	○	○

6.2.9 Addition and subtraction of floating decimal point data (E+, E+P, E-, E-P)

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J⌷\⌷		Special Function Module U⌷\G⌷	Index Register Zn	Constant E	Other
	Bit	Word		Bit	Word				
Ⓢ	–	○		–	○		–	○	–
Ⓓ	–	○		–	○		–	–	–



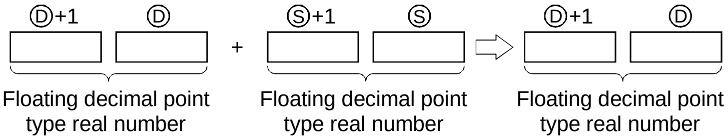
[Set Data]

Set Data	Meaning	Data Type
Ⓢ	Addition or subtraction data, or head number of device storing additon or subtractraction data	Real number
Ⓓ	Data to be added to or subtracted from, or head number of device storing such data	

[Functions]

E+

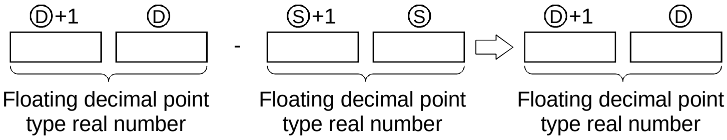
(1) Adds the floating decimal point type real number designated at Ⓓ and the floating decimal point type real number designated at Ⓢ, and stores the sum in the device designated at Ⓓ.


Ⓓ+1
Ⓓ
+
Ⓢ+1
Ⓢ
Ⓓ+1
Ⓓ
Floating decimal point type real number
Floating decimal point type real number
Floating decimal point type real number

(2) Values which can be designated at Ⓢ and Ⓓ and which can be stored, are as follows:
0, $\pm 2^{-126} \leq$ | Designated value (stored value) | $< \pm 2^{128}$

E-

(1) Subtracts a floating decimal point type real number designated by Ⓓ and a floating decimal point type real number designated by Ⓢ, and stores the result at a device designated by Ⓓ.


Ⓓ+1
Ⓓ
-
Ⓢ+1
Ⓢ
Ⓓ+1
Ⓓ
Floating decimal point type real number
Floating decimal point type real number
Floating decimal point type real number

(2) Values which can be designated at Ⓢ and Ⓓ and which can be stored, are as follows:
0, $\pm 2^{-126} \leq$ | Designated value (stored value) | $< \pm 2^{128}$

[Operation Errors]

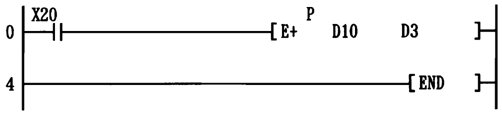
(1) In the following cases an operation error occurs, the error flag (SM0) turns ON, and an error code is stored at SD0.

- The contents of the designated device or the result of the addition or subtraction operation are not "0", or not within the following range: (Error code: 4100)
$\pm 2^{-126} \leq$ | Contents of designated device/operation result | $< \pm 2^{128}$
- When the specified device contains -0 (Q4ARCPU)
(Operation error does not occur even if -0 is stored if SM707 is turned on.) (Error code: 4100)

[Program Example]

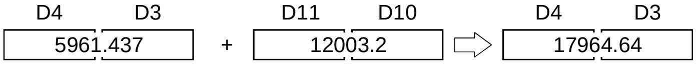
(1) The following program adds the floating decimal point type real numbers at D3 and D4 and the floating decimal point type real numbers at D10 and D11 when X20 goes ON, and stores the result at D3 and D4.

[Ladder Mode]



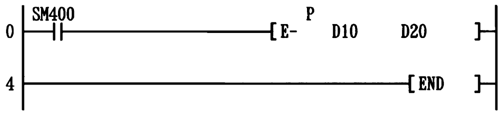
[List Mode]

Steps	Instruction	Device
0	LD	X20
1	E+P	D10 D3
4	END	



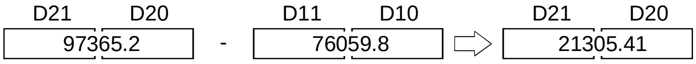
(2) The following program subtracts the floating decimal point type real number at D10 and D11 from the floating decimal point type real numbers at D20 and D21, and stores the result of the subtraction at D20 and D21.

[Ladder Mode]

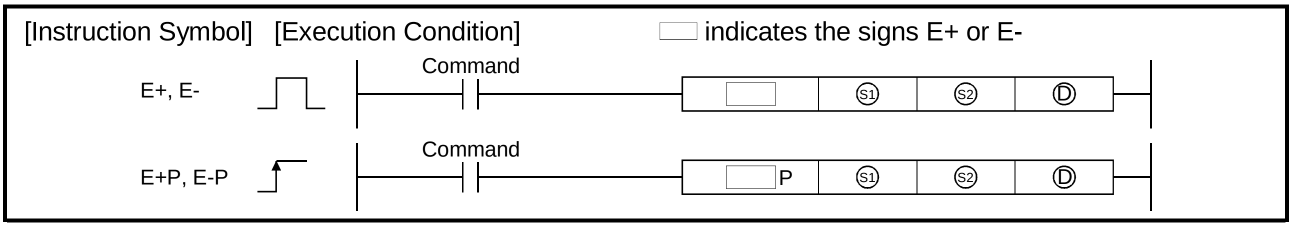


[List Mode]

Steps	Instruction	Device
0	LD	SM400
1	E-P	D10 D20
4	END	



Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant E	Other
	Bit	Word		Bit	Word				
S1	—	○		—	○		—	○	—
S2	—	○		—	○		—	○	—
D	—	○		—	○		—	—	—



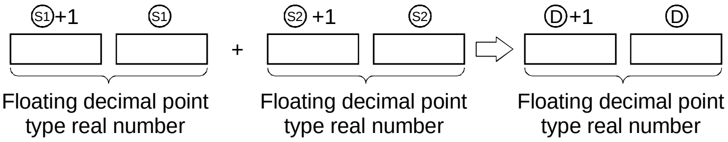
[Set Data]

Set Data	Meaning	Data Type
S1	Data to be added to or subtracted from, or the head number of the device storing such data	Real number
S2	Addition or subtraction data, or head number of device storing addition or subtraction data	
D	Head number of device storing addition or subtraction data	

[Functions]

E+

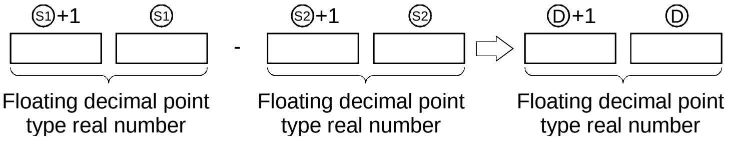
(1) Adds the floating decimal point type real number designated by S1 and the floating decimal point type real number designated by S2, and stores the result at the device designated by D.


S1+1
S1
+
S2 +1
S2
D+1
D
Floating decimal point type real number
Floating decimal point type real number
Floating decimal point type real number

(2) Values that can be designated by S1, S2 or D and values that can be stored, are as follows:
0, $\pm 2^{-126} \leq$ | Designated value (stored value) | $< \pm 2^{128}$

E-

(1) Subtracts the floating decimal point type real number designated by Ⓢ1 from the floating decimal point type real number designated by Ⓢ2, and stores the result at the device designated by Ⓓ.


Ⓢ1+1
Ⓢ1
-
Ⓢ2+1
Ⓢ2
Ⓓ+1
Ⓓ
Floating decimal point type real number
Floating decimal point type real number
Floating decimal point type real number

(2) Values that can be designated by Ⓢ1, Ⓢ2 or Ⓓ and values that can be stored, are as follows: 0, $\pm 2^{-126} \leq$ | Designated value (stored value) | < $\pm 2^{128}$

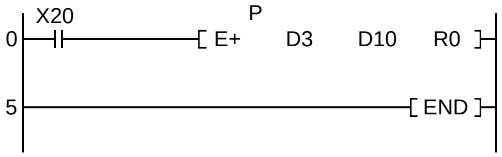
[Operation Errors]

(1) In the following cases an operation error occurs, the error flag (SM0) turns ON, and an error code is stored at SD0.
- The contents of the designated device or the result of the addition or subtraction operation are not "0", or not within the following range: (Error code: 4100)
  0, $\pm 2^{-126} \leq$ | Contents of designated device/ result of operation | < $\pm 2^{128}$
- When the specified device contains -0 (Q4ARCPU)
  (Operation error does not occur even if -0 is stored if SM707 is turned on.) (Error code: 4100)

[Program Example]

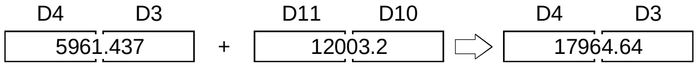
(1) The following program adds the floating decimal point type real numbers at D3 and D4 and the floating decimal point type real numbers at D10 and D11 when X20 goes ON, and outputs the result to R0 and R1.

[Ladder Mode]



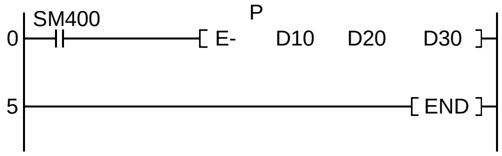
[List Mode]

Steps	Instruction	Device
0	LD	X20
1	E+P	D3 D10 R0
5	END	



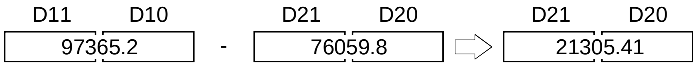
(2) The following programs subtracts the floating decimal point type real numbers at D20 and D21 from the floating decimal point type real numbers at D11 and D10, and stores the result at D30 and D31.

[Ladder Mode]



[List Mode]

Steps	Instruction	Device
0	LD	SM400
1	E-P	D10 D20 D30
5	END	

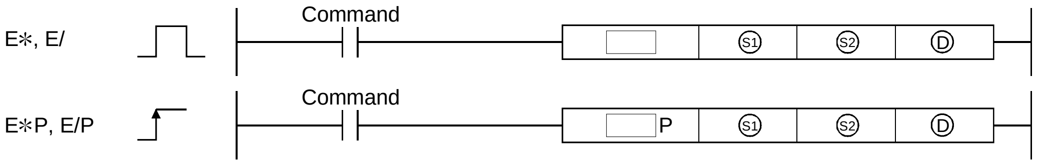


QCPU			QnA	Q4AR
PLC CPU		Process CPU		
Basic	High Performance			
×	○	○	○	○

6.2.10 Multiplication and division of floating decimal point data (E∗, E∗P, E/, E/P)

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant E	Other
	Bit	Word		Bit	Word				U
Ⓢ1	–	○	○	–	○	○	–	○	–
Ⓢ2	–	○	○	–	○	○	–	○	–
Ⓓ	–	○	○	–	○	○	–	–	–

[Instruction Symbol] [Execution Condition] □ indicates the signs E∗ or E/



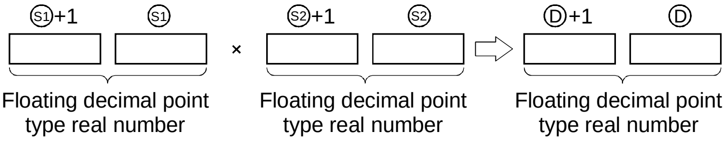
[Set Data]

Set Data	Meaning	Data Type
Ⓢ1	Data that will be multiplied or divided, or the head number of the device storing data that will be multiplied or divided	Real number
Ⓢ2	Data to multiply or divide by, or the head number of device storing such data	
Ⓓ	Head number of the device storing the operation results of multiplication or division operation	

[Functions]

E∗

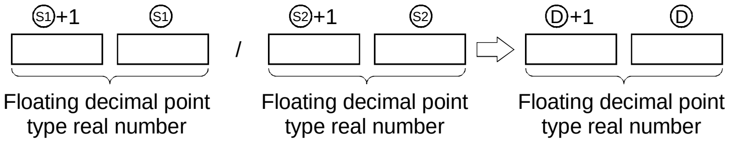
(1) Multiplies the floating decimal point type real numbers designated by Ⓢ1 and the floating decimal point type real numbers designated by Ⓢ2, and stores the product at the device designated by Ⓓ.


Ⓢ1+1
Ⓢ1
×
Ⓢ2+1
Ⓢ2
Ⓓ+1
Ⓓ
Floating decimal point type real number
Floating decimal point type real number
Floating decimal point type real number

(2) Values that can be designated by Ⓢ1, Ⓢ2 or Ⓓ, and values that can be stored, are as follows:
$0, \pm 2^{-126} \leq |$ Designated value (stored value) $| < \pm 2^{128}$

E/

(1) Divides floating decimal point type real numbers designated by Ⓢ1 by floating decimal point type real numbers designated by Ⓢ2, and stores the result in the device designated by Ⓓ.


Ⓢ1+1
Ⓢ1
/
Ⓢ2+1
Ⓢ2
Ⓓ+1
Ⓓ
Floating decimal point type real number
Floating decimal point type real number
Floating decimal point type real number

(2) Values that can be designated by Ⓢ1, Ⓢ2 or Ⓓ, and values that can be stored, are as follows:
0, $\pm 2^{-126} \leq$ | Designated value (stored value) | $< \pm 2^{128}$

[Operation Errors]

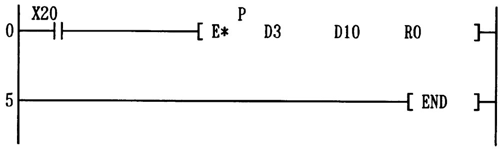
(1) In the following cases an operation error occurs, the error flag (SM0) turns ON, and an error code is stored at SD0.

- The contents of the designated device or the result of the multiplication or division operation are not "0", or not within the following range: (Error code: 4100)
  0, $\pm 2^{-126} \leq$ | Contents of designated device/results of operation | $< \pm 2^{128}$
- When the specified device contains -0 (Q4ARCPU)
  (Operation error does not occur even if -0 is stored if SM707 is turned on.) (Error code: 4100)

[Program Example]

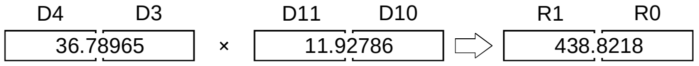
(1) The following program multiplies the floating decimal point real numbers at D3 and D4 and the floating decimal point real numbers at D10 and D11, and stores the result at R0 and R1.

[Ladder Mode]



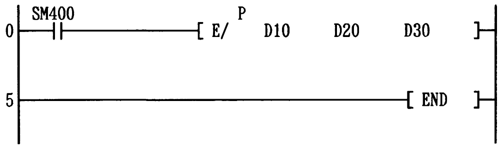
[List Mode]

Steps	Instruction	Device
0	LD	X20
1	E*P	D3 D10 R0
5	END	



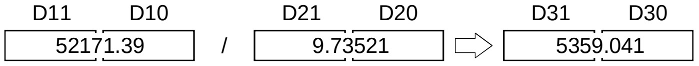
(2) The following program divides the floating decimal point real numbers at D10 and D11 by the floating decimal point real numbers at D20 and D21, and stores the result at D30 and D31.

[Ladder Mode]



[List Mode]

Steps	Instruction	Device
0	LD	SM400
1	E/P	D10 D20 D30
5	END	

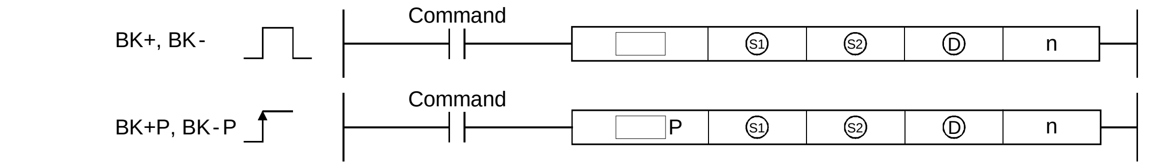


QCPU			QnA	Q4AR
PLC CPU		Process CPU		
Basic	High Performance			
○	○	○	○	○

6.2.11 Block addition and subtraction (BK+, BK+P, BK-, BK-P)

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant K, H	Other
	Bit	Word		Bit	Word				
Ⓢ1	—	○		—				—	—
Ⓢ2	—	○		—				○	—
Ⓓ	—	○		—				—	—
n	○	○		○				○	—

[Instruction Symbol] [Execution Condition] □ indicates the signs BK+ or BK-



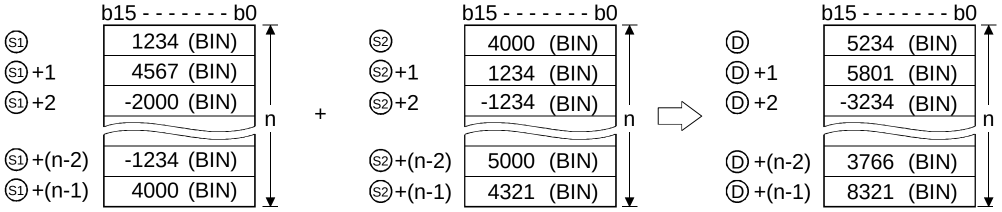
[Set Data]

Set Data	Meaning	Data Type
Ⓢ1	Data to be added to or subtracted from, or head number of device storing such data	BIN 16 bits
Ⓢ2	Addition or subtraction data, or head number of device storing addition or subtraction data	
Ⓓ	Head number of the devices where the operation results are stored	
n	Number of addition/subtraction data blocks	

[Functions]

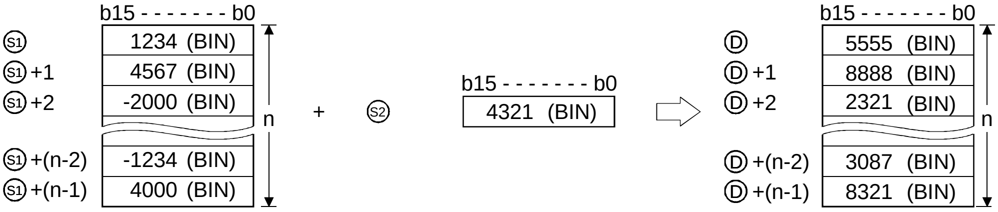
BK+

(1) Adds n-points of BIN data from the device designated by Ⓢ1 and n-points of BIN data from the device designated by Ⓢ2 and stores the result from the device designated by Ⓓ onward.



(2) Block addition is performed in 16-bit units.

(3) The constant designated by (S2) can be between -32768 to 32767 (BIN 16 bits).

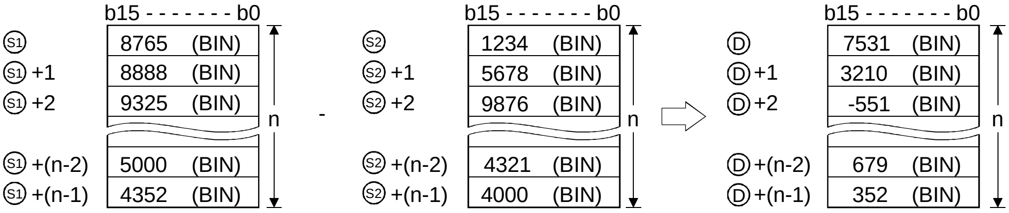


(4) The following happens if an underflow or overflow is generated in the operation results: The carry flag in this case does not go ON.

- K32767 +K2 → K-32767
  (H7FFF) (H0002) (H8001)
- K-32767 +K-2 → K32766
  (H8000) (HFFFE) (H7FFE)

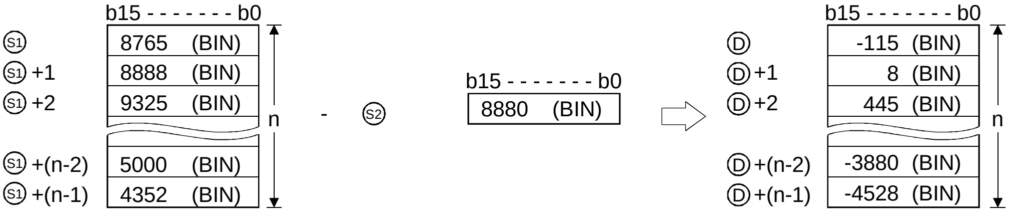
BK–

(1) Subtracts n-points of BIN data from the device designated by (S1) and n-points of BIN data from the device designated by (S2) and stores the result from the device designated by (D) onward.



(2) Block subtraction is performed in 16-bit units.

(3) The constant designated by (S2) can be between-32768 to 32767 (BIN 16 bits).



(4) The following happens if an underflow or overflow is generated in the operation results: The carry flag in this case does not go ON.

- K-32768 -K2 → K32766
  (H8000) (H0002) (H7FFF)
- K32767 -K-2 → -32766
  (H8000) (H0002) (H8001)

[Operation Errors]

(1) In the following cases an operation error occurs, the error flag (SM0) turns ON, and an error code is stored at SD0.

- The n-bit range from the (S1), (S2), or (D) device exceeds the range of that device.
- The device range for n points starting from the device designated by (S1) overlaps with the device range for n points starting from the device designated by (D). (Error code: 4101)
- The device range for n points starting from the device designated by (S2) overlaps with the device range for n points starting from the device designated by (D). (Error code: 4101)
- The device range for n points starting from the device designated by (S1) overlaps with the device range for n points starting from the device designated by (S2). (Error code: 4101)

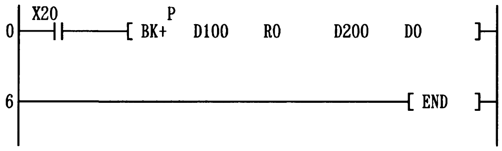
[Program Example]

(1) When X20 is ON, the program performs additions of the following data:

- The data in the number (value stored in D0) of devices starting from D100
- The data in the number (value stored in D0) of devices starting from R100

Then the program stores the addition results at the number (value stored in D0) of devices starting from D200.

[Ladder Mode]



[List Mode]

Steps	Instruction	Device
0	LD	X20
1	BK+P	D100 R0 D200 D0
6	END	

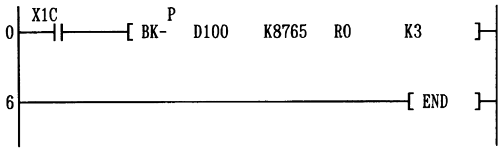
	b15 - - - - - - - b0			b15 - - - - - - - b0			b15 - - - - - - - b0
D100	6789 (BIN)	+	R0	1234 (BIN)	⇨	D200	8023 (BIN)
D101	7821 (BIN)		R1	2032 (BIN)		D201	9853 (BIN)
D102	5432 (BIN)		R2	-3252 (BIN)		D202	2180 (BIN)
D103	3520 (BIN)		R3	-1000 (BIN)		D203	2520 (BIN)

D0	4

(2) When X1C is ON, the following program subtracts the constant 8765 from the data in three devices starting from D100.

Then the program stores the subtraction results at the number of the three devices starting from R0

[Ladder Mode]



[List Mode]

Steps	Instruction	Device
0	LD	X1C
1	BK-P	D100 K8765 R0 K3
5	END	

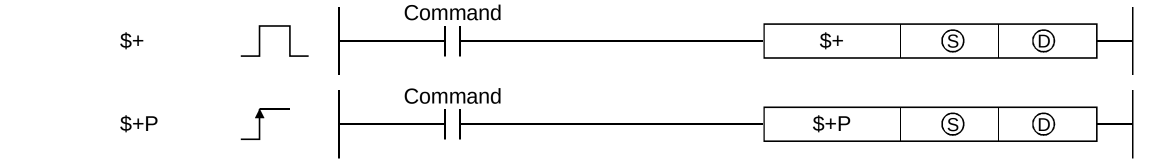
	b15 - - - - - - - b0		b15 - - - - - - - b0			b15 - - - - - - - b0
D100	12345 (BIN)	-	8765 (BIN)	⇨	R0	3580 (BIN)
D101	8701 (BIN)				R1	-64 (BIN)
D102	3502 (BIN)				R2	-5263 (BIN)

QCPU			QnA	Q4AR
PLC CPU		Process CPU		
Basic	High Performance			
×	○	○	○	○

6.2.12 Linking character strings ($+, $+P)

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant $	Other
	Bit	Word		Bit	Word				
Ⓢ	—	○		—				○	—
Ⓓ	—	○		—				—	—

[Instruction Symbol] [Execution Condition]



[Set Data]

Set Data	Meaning	Data Type
Ⓢ	Head number of device holding linked data or data	Character string
Ⓓ	Head number of device holding data which has been linked	

[Functions]

(1) Character string data stored in device numbers starting with that designated at Ⓢ will be appended after character string data stored in device numbers starting with that designated at Ⓓ, and will be stored in device numbers starting with that designated at Ⓓ.
The object of character string data is that character string data stored from device numbers designated at Ⓓ and Ⓢ to that stored at "00H".

	b15 - - - b8	b7 - - - - b0
Ⓓ	42H (B)	41H (A)
Ⓓ +1	44H (D)	43H (C)
Ⓓ +2	00H	45H (E)

"ABCDE"

+

	b15 - - - b8	b7 - - - - b0
Ⓢ	32H (2)	31H (1)
Ⓢ +1	34H (4)	33H (3)
Ⓢ +2	36H (6)	35H (5)
Ⓢ +3	00H	

"123456"

⇨

	b15 - - - b8	b7 - - - - b0
Ⓓ	42H (B)	41H (A)
Ⓓ +1	44H (D)	43H (C)
Ⓓ +2	31H (1)	45H (E)
Ⓓ +3	33H (2)	32H (2)
Ⓓ +4	35H (3)	34H (4)
Ⓓ +5	00H	36H (6)

"ABCDE123456"

(2) When character strings are linked, the "00H", which indicates the end of character string data designated at Ⓓ, is ignored, and the character string designated at Ⓢ is appended to the last character of the Ⓓ string.

[Operation Errors]

(1) In the following cases an operation error occurs, the error flag (SM0) turns ON, and an error code is stored at SD.

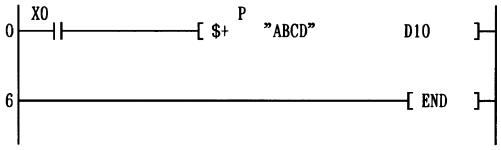
- The entire character string linked from the device number designated by Ⓓ to the final device number of the relevant device cannot be stored. (Error code: 4100)
- The storage device numbers for the character strings designated by Ⓢ and Ⓓ overlap. (Error code: 4101)

(2) See Section 3.6 for information regarding errors during index modification.

[Program Example]

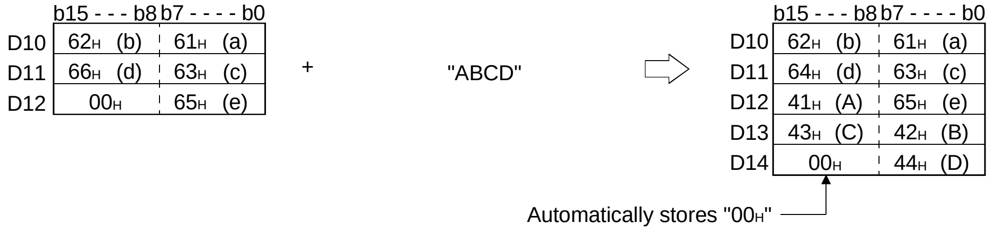
(1) The following program links the character string stored from D10 to D12 to the character string "ABCD" when X0 is ON.

[Ladder Mode]



[List Mode]

Steps	Instruction	Device
0	LD	X0
1	$+P	"ABCD" D10
6	END	



	b15 - - - b8	b7 - - - - b0
D10	62H (b)	61H (a)
D11	66H (d)	63H (c)
D12	00H	65H (e)

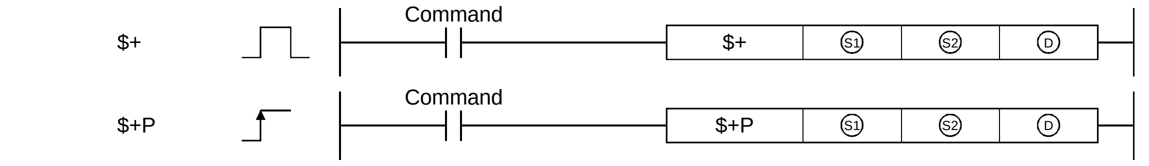
+ "ABCD" ⇨

	b15 - - - b8	b7 - - - - b0
D10	62H (b)	61H (a)
D11	64H (d)	63H (c)
D12	41H (A)	65H (e)
D13	43H (C)	42H (B)
D14	00H	44H (D)

Automatically stores "00H"

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant $	Other
	Bit	Word		Bit	Word				
(S1)	–	○		–				○	–
(S2)	–	○		–				○	–
(D)	–	○		–				–	–

[Instruction Symbol] [Execution Condition]



[Set Data]

Set Data	Meaning	Data Type
(S1)	Head number of device holding linked data	Character string
(S2)	Head number of device holding data which has been linked	
(D)	Head number of device holding results of linking	

[Functions]

(1) Appends character string data stored from the device number designated by (S2) to the character string data stored from the device number designated by (S1), and stores it from the device number designated by (D1).

	b15 - - - b8	b7 - - - - b0
(S1)	46H (F)	48H (H)
(S1) +1	2DH (-)	41H (A)
(S1) +2	00H	

+

	b15 - - - b8	b7 - - - - b0
(S2)	35H (5)	31H (1)
(S2) +1	39H (9)	33H (3)
(S2) +2	00H	41H (A)

⇨

	b15 - - - b8	b7 - - - - b0
(D)	46H (F)	48H (H)
(D) +1	2DH (-)	41H (A)
(D) +2	35H (5)	31H (1)
(D) +3	39H (9)	33H (3)
(D) +4	00H	41H (A)

(2) When character strings are linked, the "00H" which indicates the end of character string data indicated by (S1), is ignored, and the character string indicated by (S2) is appended to the last character of the (S1) string.

[Operation Errors]

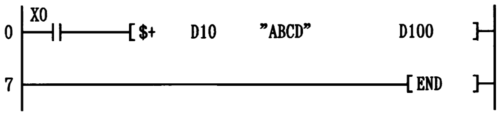
(1) In the following cases an operation error occurs and the error flag goes ON.

- The entire character string linked from the device number designated by Ⓓ to the final device number of the relevant device cannot be stored.
- The storage device numbers for the character strings designated by Ⓢ1 or Ⓢ2 overlap with those for Ⓓ.

[Program Example]

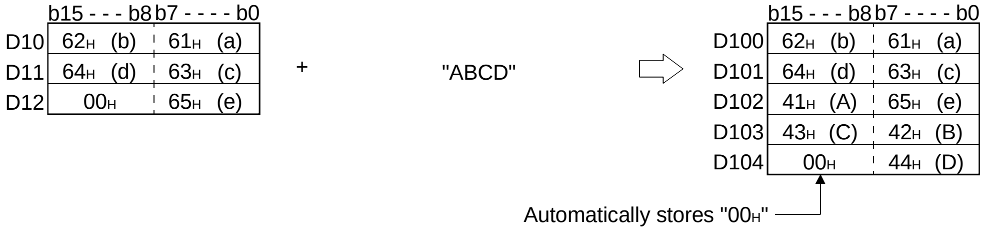
(1) The following program links the character string stored from D10 to D12 with the character string "ABCD" when X0 is ON, and stores them in D100 onwards.

[Ladder Mode]



[List Mode]

Steps	Instruction	Device
0	LD	X0
1	$+	D10 "ABCD" D100
7	END	



	b15 - - - b8	b7 - - - - b0
D10	62H (b)	61H (a)
D11	64H (d)	63H (c)
D12	00H	65H (e)

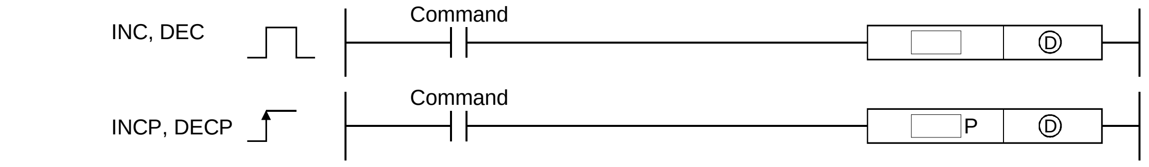
	b15 - - - b8	b7 - - - - b0
D100	62H (b)	61H (a)
D101	64H (d)	63H (c)
D102	41H (A)	65H (e)
D103	43H (C)	42H (B)
D104	00H	44H (D)

QCPU				QnA	Q4AR
PLC CPU		Process CPU			
Basic	High Performance				
○	○	○		○	○

6.2.13 Incrementing and decrementing 16-bit BIN data (INC, INCP, DEC, DECP)

Set Data	Usable Devices									
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant K, H	Other
	Bit	Word		Bit	Word				
Ⓓ	○							—	

[Instruction Symbol] [Execution Condition] □ indicates INC or DEC



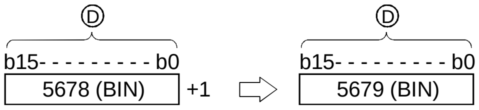
[Set Data]

Set Data	Meaning	Data Type
Ⓓ	Head number of device conducting INC (add 1) or DEC (subtract 1) operation	BIN 16 bits

[Functions]

INC

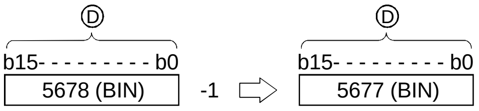
(1) Adds 1 to device designated by Ⓓ (16-bit data).



(2) If the contents of the device designated by Ⓓ were 32767, and the INC or INCP instruction were executed on that device, the value -32768 would be stored in the device designated by Ⓓ.

DEC

(1) Subtracts 1 from device designated by Ⓓ (16-bit data).



(2) If the contents of the device designated by Ⓓ were 0, and the DEC or DECP instruction were executed on that device, the value -1 would be stored in the device designated by Ⓓ.

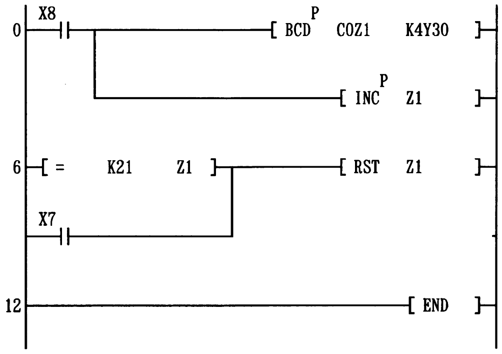
[Operation Errors]

(1) There are no operation errors associated with the INC, INCP, DEC or DECP instructions.

[Program Example]

(1) The present value stored in counter C0 to C20 is output to Y30 to Y3F as BCD data when X8 is on. (When present value is less than 9999)

[Ladder Mode]



Outputs the present value of C (D+Z1) to Y30 to Y3F as BCD.

Executes Z1+1

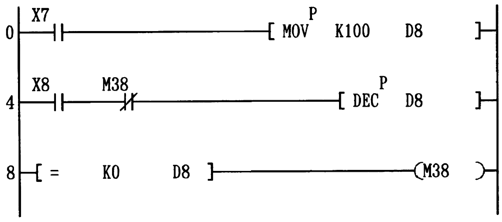
With Z1=21 or X7 (reset input), sets Z to 0

[List Mode]

Steps	Instruction	Device
0	LD	X8
1	BCDP	C0Z1
		K4Y30
4	INCP	Z1
6	LD=	K21
		Z1
9	OR	X7
10	RST	Z1
12	END	

(2) The following is a down counter program.

[Ladder Mode]



Transfers the value 100 to D8 when X7 is ON

When M38 is OFF, X8 goes from OFF to ON, and 1 is decremented from D8.

At D8=0, M38 goes ON

[List Mode]

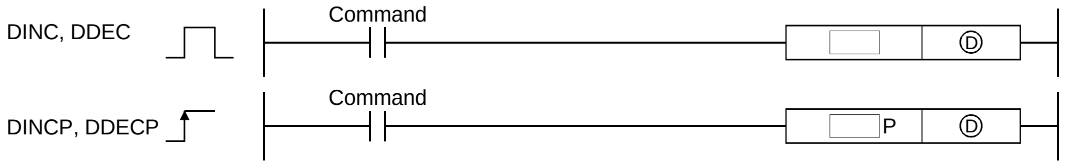
Steps	Instruction	Device
0	LD	X7
1	MOVP	K100
		D8
4	LD	X8
5	ANI	M38
6	DECP	D8
8	LD=	K0
		D8
11	OUT	M38
12	END	

QCPU			QnA	Q4AR
PLC CPU		Process CPU		
Basic	High Performance			
○	○	○	○	○

6.2.14 Incrementing and decrementing 32-bit BIN data (DINC, DINCP, DDEC, DDECP)

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant K, H	Other
	Bit	Word		Bit	Word				
Ⓓ	○							—	

[Instruction Symbol] [Execution Condition] □ indicates DINC/DDEC



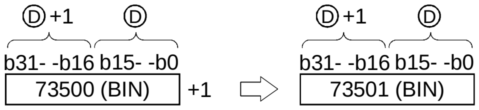
[Set Data]

Set Data	Meaning	Data Type
Ⓓ	Head number of device what will execute the DINC (+1) or DDEC (-1) operation	BIN 32 bits

[Functions]

DINC

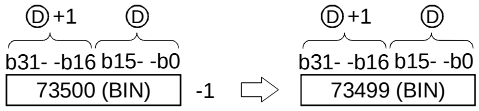
(1) Adds 1 to the device designated by Ⓓ (32-bit data).



(2) If the contents of the device designated by Ⓓ are 2147483647, and the DINC or DINCP instruction is executed, the value -2147483648 will be stored at the device designated by Ⓓ.

DDEC

(1) Subtracts 1 from the device designated by Ⓓ (32-bit data).



(2) If the contents of the device designated by Ⓓ are 0, and the DINC or DINCP instruction is executed, the value -1 will be stored at the device designated by Ⓓ.

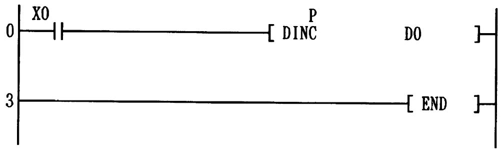
[Operation Errors]

(1) There are no operation errors associated with DINC(P) or DDEC(P).

[Program Example]

(1) The following program adds 1 to the data at D0 and D1 when X0 is ON.

[Ladder Mode]

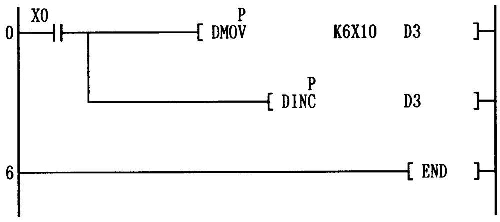


[List Mode]

Steps	Instruction	Device
0	LD	X0
1	DINCP	D0
3	END	

(2) The following program adds 1 to the data set at X10 to X27 when X0 goes ON, and stores the result at D3 and D4.

[Ladder Mode]

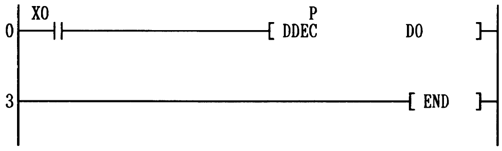


[List Mode]

Steps	Instruction	Device
0	LD	X0
1	DMOVP	K6X10 D3
4	DINCP	D3
6	END	

(3) The following program subtracts 1 from the data at D0 and D1 when X0 goes ON.

[Ladder Mode]

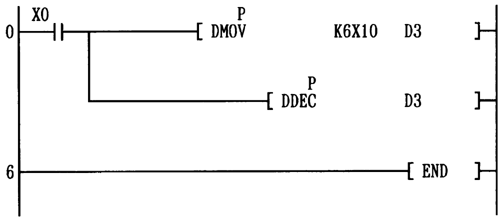


[List Mode]

Steps	Instruction	Device
0	LD	X0
1	DDECP	D0
3	END	

(4) The following program subtracts 1 from the data set at X10 to X27 when X0 goes ON, and stores the result at D3 and D4.

[Ladder Mode]



[List Mode]

Steps	Instruction	Device
0	LD	X0
1	DMOVP	K6X10 D3
4	DDECP	D3
6	END	

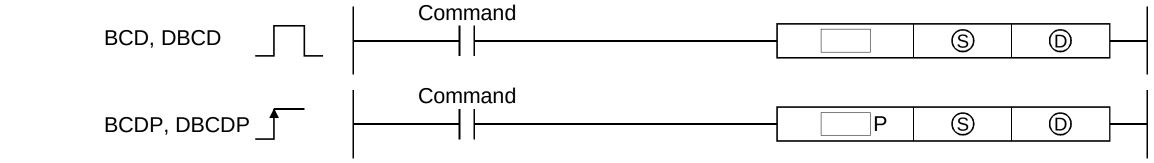
QCPU			QnA	Q4AR
PLC CPU		Process CPU		
Basic	High Performance			
○	○	○	○	○

6.3 Data Conversion Instructions

6.3.1 Conversion from BIN data to 4-digit and 8-digit BCD (BCD, BCDP, DBCD, DBCDP)

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant K, H	Other
	Bit	Word		Bit	Word				
Ⓢ	○							○	–
Ⓓ	○							–	–

[Instruction Symbol] [Execution Condition] □ indicates BCD or DBCD



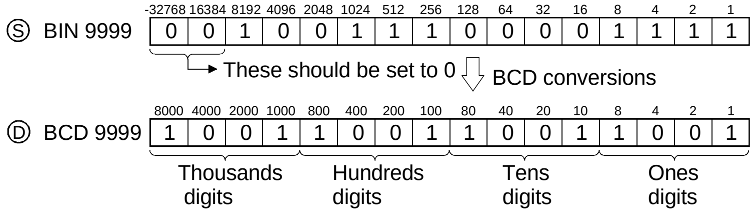
[Set Data]

Set Data	Meaning	Data Type
Ⓢ	BIN data, or head number of the device where BIN data is stored	BIN 16/ 32 bits
Ⓓ	Head number of the device that will store BCD data	BCD 4/8 digits

[Functions]

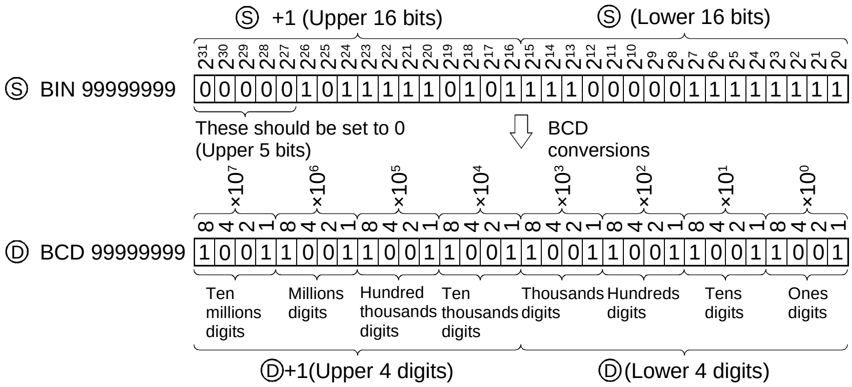
BCD

Converts BIN data (0 to 9999) at the device designated by Ⓢ to BCD data, and stores it at the device designated by Ⓓ.



DBCD

Converts BIN data (0 to 99999999) at the device designated by Ⓢ to BCD data, and stores it at the device designated by Ⓓ.



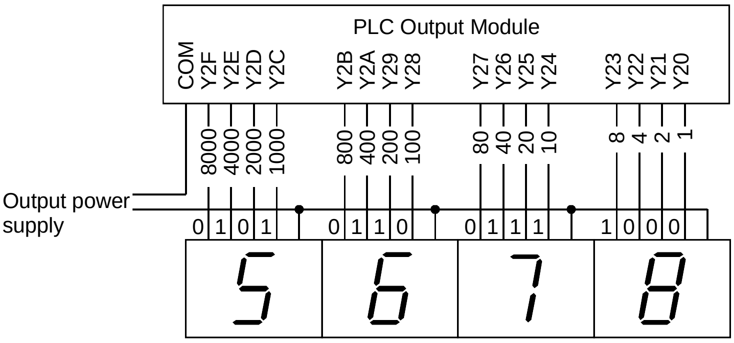
[Operation Errors]

(1) In the following cases an operation error occurs, the error flag (SM0) turns ON, and an error code is stored at SD0.

- The data at Ⓢ was not in the 0 to 9999 range when the BCD instruction was issued. (Error code: 4100)
- The data at Ⓢ+1 and Ⓢ was not in the 0 to 99999999 range when the DBCD instruction was issued. (Error code: 4100)

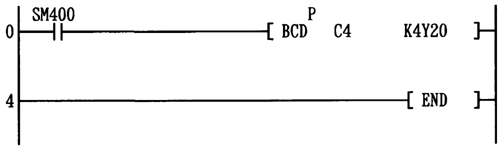
[Program Example]

(1) The following program outputs the present value of C4 from Y20 to Y2F to the BCD display device.



7-element display unit

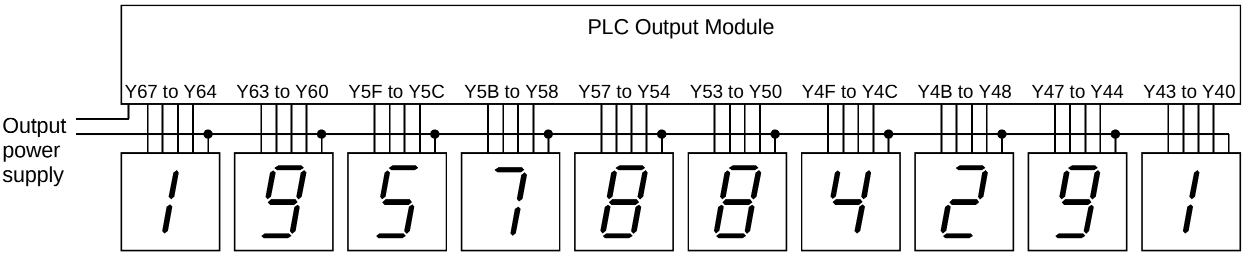
[Ladder Mode]



[List Mode]

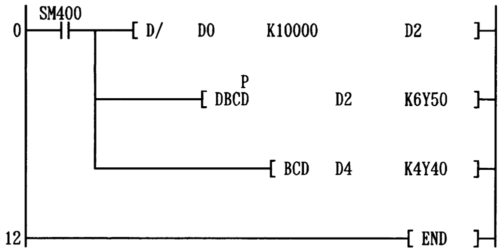
Steps	Instruction	Device
0	LD	SM400
1	BCDP	C4 K4Y20
4	END	

(2) The following program outputs 32-bit data from D0 to D1 to Y40 to Y67.



7-element display unit

[Ladder Mode]



[List Mode]

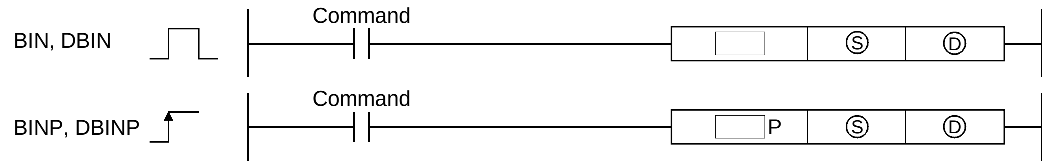
Steps	Instruction	Device
0	LD	SM400
1	D/	D0 K10000 D2
6	DBCDP	D2 K6Y50
9	BCD	D4 K4Y40
12	END	

QCPU			QnA	Q4AR
PLC CPU		Process CPU		
Basic	High Performance			
○	○	○	○	○

6.3.2 Conversion from BCD 4-digit and 8-digit data to BIN data (BIN, BINP, DBIN, DBINP)

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant K, H	Other
	Bit	Word		Bit	Word				U
Ⓢ	○							○	–
Ⓓ	○							–	–

[Instruction Symbol] [Execution Condition] □ indicates BIN or DBIN



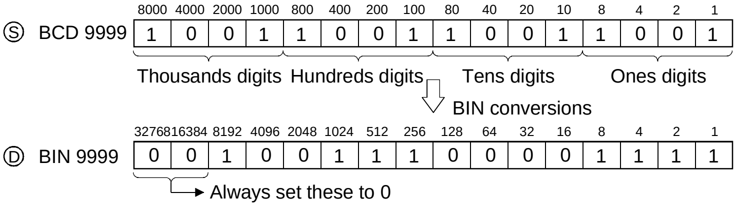
[Set Data]

Set Data	Meaning	Data Type
Ⓢ	BCD data or head number of device storing BCD data	BCD 4/8 digits
Ⓓ	Head number of device that will store BIN data	BIN 16/32 bits

[Functions]

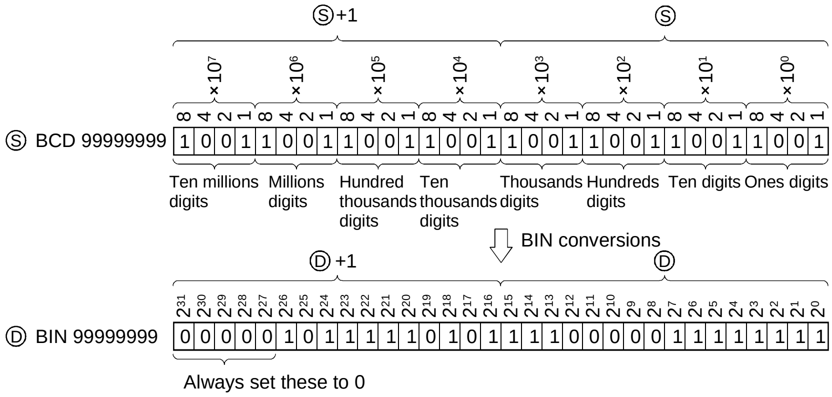
BIN

Converts BCD data (0 to 9999) at device designated by Ⓢ to BIN data, and stores at the device designated by Ⓓ.



DBIN

Converts BCD data (0 to 99999999) at device designated by Ⓢ to BIN data, and stores at the device designated by Ⓓ.



[Operation Errors]

(1) In the following cases, an operation error occurs, the error flag (SM0) turns ON, an error code is stored in SD0, and the instruction is not executed.

- When values other than 0 to 9 are designated to any digits of Ⓢ.

[When QCPU is used]

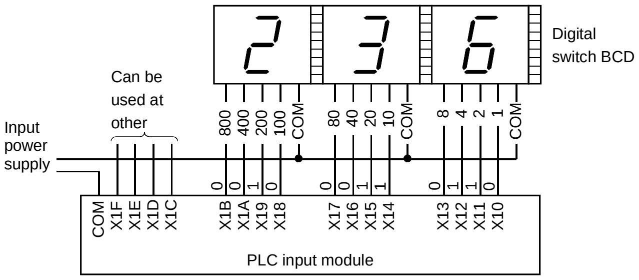
When QCPU is used, the error above can be suppressed by turning ON SM722.

However, the instruction is not executed regardless of whether SM722 is turned ON or OFF if the designated value is out of the available range.

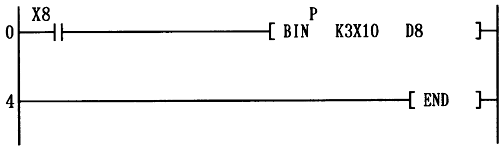
For BCDP/DBCDP instructions, the next operation is disabled regardless of the presence of errors unless the execution condition is turned from OFF to ON.

[Operation Example]

(1) The following program converts the BCD data at X10 to X1B to BIN when X8 is ON, and stores it at D8.



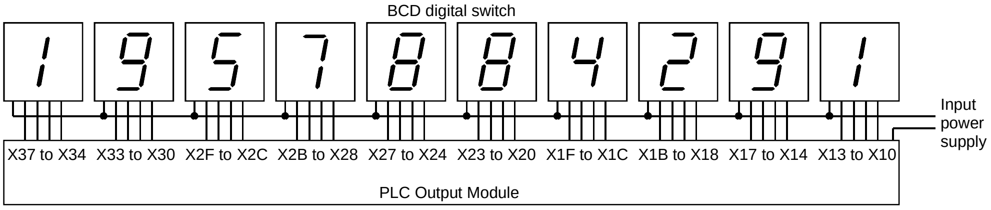
[Ladder Mode]



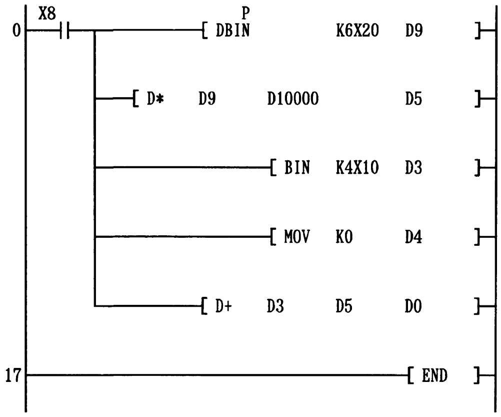
[List Mode]

Steps	Instruction	Device
0	LD	X8
1	BINP	K3X10 D8
4	END	

(2) The following program converts the BCD data at X10 to X37 to BIN when X8 is ON, and stores it at D0 and D1.



[Ladder Mode]



[List Mode]

Steps	Instruction	Device
0	LD	X8
1	DBINP	K6X20 D9
4	D*	D9 D10000 D5
8	BIN	K4X10 D3
11	MOV	K0 D4
13	D+	D3 D5 D0
17	END	

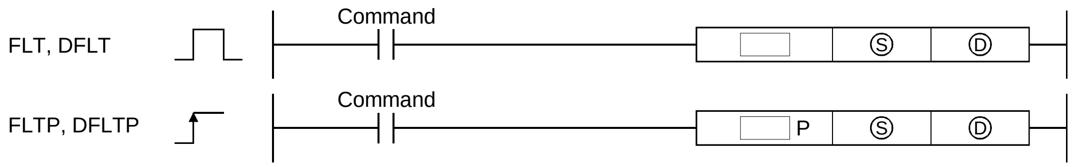
If the data set at X10 to X37 is a BCD value which exceeds 2147483647, the value at D0 and D1 will be a negative value, because it exceeds the range of numerical values that can be handled by a 32-bit device.

QCPU			QnA	Q4AR
PLC CPU		Process CPU		
Basic	High Performance			
×	○	○	○	○

6.3.3 Conversion from BIN 16 and 32-bit data to floating decimal point (FLT, FLTP, DFLT, DFLTP)

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant K, H	Other
	Bit	Word		Bit	Word				
Ⓢ	○	○		○	○		○		–
Ⓓ	–	○		–	○		–		–

[Instruction Symbol] [Execution Condition] □ indicates FLT or DFLT



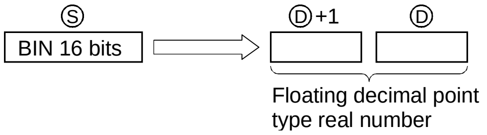
[Set Data]

Set Data	Meaning	Data Type
Ⓢ	Head device number where integer data for the purpose of conversion to floating decimal point data is being stored	BIN 16/32 bits
Ⓓ	Head device number that will store converted floating decimal point data	Real number

[Functions]

FLT

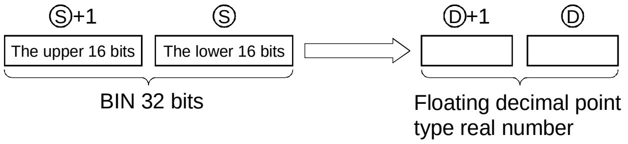
(1) Converts 16-bit BIN data designated by Ⓢ to floating decimal point type real number, and stores at device number designated by Ⓓ.



(2) BIN values between -32768 to 32767 can be designated by Ⓢ.

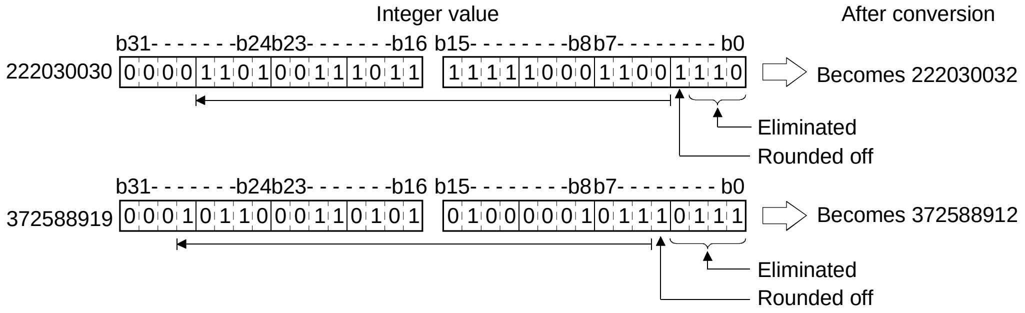
DFLT

(1) Converts 32-bit BIN data designated by Ⓢ to floating decimal point type real number, and stores at device number designated by Ⓓ.



(2) BIN values between -2147483648 to 2147483647 can be designated by Ⓢ+1 and Ⓢ.

(3) Due to the fact that floating decimal point type real numbers are processed by simple 32-bit processing, the number of significant digits is 24 bits if the display is binary and approximately 7 digits if the display is decimal.
For this reason, if the integer exceeds the range of -16777216 to 16777215 (24-bit BIN value), errors can be generated in the conversion value.
The conversion results round off at the 25th bit from the highest bit of the integer value, and eliminate everything from the 26th bit and beyond.



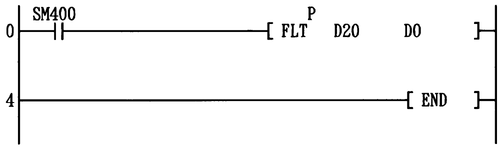
[Operation Errors]

(1) There are no errors associated with the FLT (P) or DFLT (P) instructions.

[Program Example]

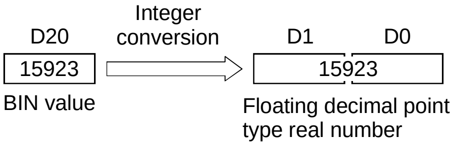
(1) The following program converts the BIN 16-bit data at D20 to a floating decimal point type real number and stores the result at D0 and D1.

[Ladder Mode]



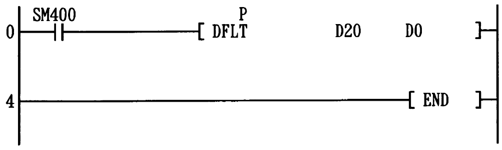
[List Mode]

Steps	Instruction	Device
0	LD	SM400
1	FLTP	D20 D0
4	END	



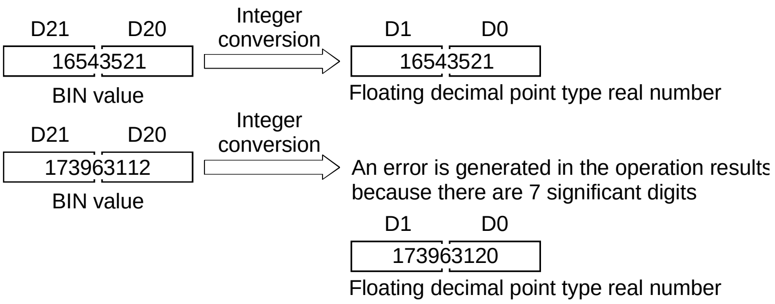
(2) The following program converts the BIN 32-bit data at D20 and D21 to a floating decimal point type real number, and stores the result at D0 and D1.

[Ladder Mode]



[List Mode]

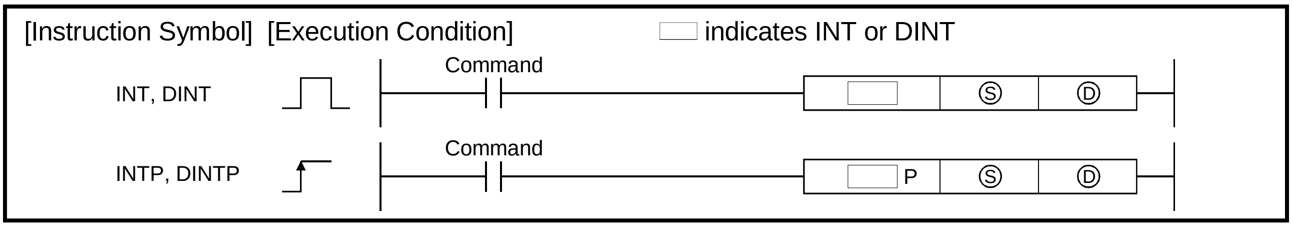
Steps	Instruction	Device
0	LD	SM400
1	DFLTP	D20 D0
4	END	


D21
D20
16543521
BIN value
Integer conversion
D1
D0
16543521
Floating decimal point type real number
D21
D20
173963112
BIN value
Integer conversion
An error is generated in the operation results because there are 7 significant digits
D1
D0
173963120
Floating decimal point type real number

QCPU			QnA	Q4AR
PLC CPU		Process CPU		
Basic	High Performance			
×	○	○	○	○

6.3.4 Conversion from floating decimal point data to BIN 16- and 32-bit data (INT, INTP, DINT, DINTP)

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant E	Other
	Bit	Word		Bit	Word				
Ⓢ	–	○		–	○		–	○	–
Ⓓ	○	○		○	○		○	–	–



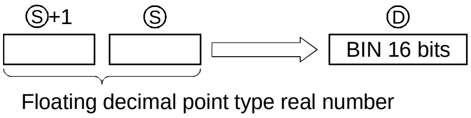
[Set Data]

Set Data	Meaning	Data Type
Ⓢ	Head device number storing floating decimal point data that will be converted to BIN value	Real number
Ⓓ	Head device number to store BIN value after conversion	BIN 16/32 bits

[Functions]

INT

(1) Converts the floating decimal point real number designated at Ⓢ into BIN 16-bit data and stores it at the device number designated at Ⓓ.



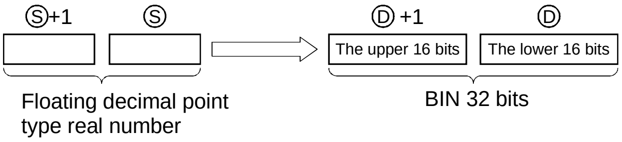
(2) The range of floating decimal point type real numbers that can be designated at Ⓢ+1 or Ⓢ is from -32768 to 32767.

(3) Stores integer values stored at Ⓓ as BIN 16-bit values.

(4) After conversion, the first digit after the decimal point of the real number is rounded off.

DINT

(1) Converts floating decimal point type real number designated by Ⓢ to BIN 32-bit data, and stores the result at the device number designated by Ⓓ.



(2) The range of floating decimal point type real numbers that can be designated at Ⓢ+1 or Ⓢ is from -2147483648 to 2147483647.

(3) The integer value stored at Ⓓ+1 and Ⓓ is stored as BIN 32 bits.

(4) After conversion, the first digit after the decimal point of the real number is rounded off.

[Operation Errors]

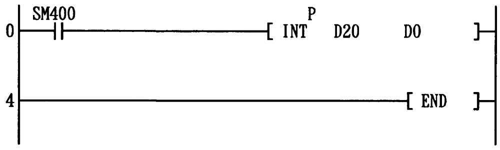
(1) In the following cases an operation error occurs, the error flag (SM0) turns ON, and an error code is stored at SD0.

- The floating decimal point type data designated by Ⓢ when the INT instruction was used was outside the -31768 to 32767 range.
- The floating decimal point type data designated by Ⓢ when the DINT instruction was used was outside the -2147483648 to 2147483647 range.

[Program Example]

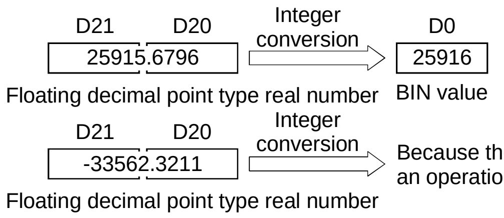
(1) The following program converts the floating decimal point type real number at D20 and D21 to BIN 16-bit data, and stores the result at D0.

[Ladder Mode]



[List Mode]

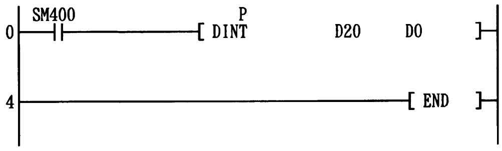
Steps	Instruction	Device
0	LD	SM400
1	INTP	D20 D0
4	END	



Because the set data is less than -32768, an operation error is returned

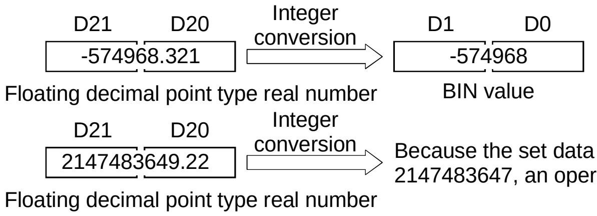
(2) The following program converts the floating decimal point type real number at D20 and D21 to BIN 32-bit data and stores the result at D0 and D1.

[Ladder Mode]



[List Mode]

Steps	Instruction	Device
0	LD	SM400
1	DINTP	D20 D0
4	END	

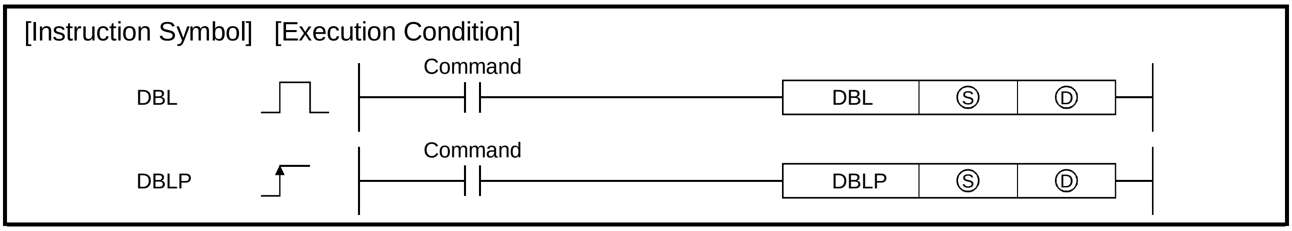


Because the set data is larger than 2147483647, an operation error is returned.

QCPU			QnA	Q4AR
PLC CPU		Process CPU		
Basic	High Performance			
○	○	○	○	○

6.3.5 Conversion from BIN 16-bit to BIN 32-bit data (DBL, DBLP)

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant K, H	Other
	Bit	Word		Bit	Word				
Ⓢ	○							○	–
Ⓓ	○							–	–

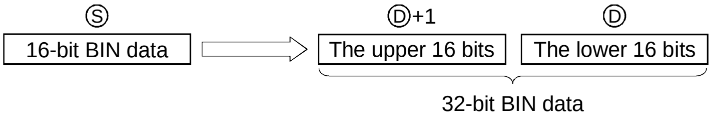


[Set Data]

Set Data	Meaning	Data Type
Ⓢ	Head number of device where BIN 16-bit data is stored	BIN 16 bits
Ⓓ	Head number of device where BIN 32-bit data is stored after conversion	BIN 32 bits

[Functions]

Converts BIN 16-bit data at device designated by Ⓢ to BIN 32-bit data with sign, and stores the result at a device designated by Ⓓ.



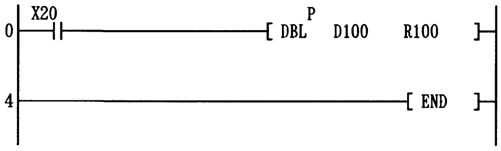
[Operation Errors]

(1) There are no errors associated with the DBL(P) instruction.

[Program Example]

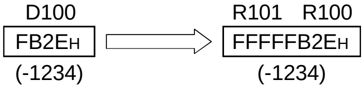
(1) The following program converts the BIN 16-bit data stored at D100 to BIN 32-bit data when X20 is ON, and stores at R100 and R101.

[Ladder Mode]



[List Mode]

Steps	Instruction	Device
0	LD	X20
1	DBLP	D100 R100
4	END	

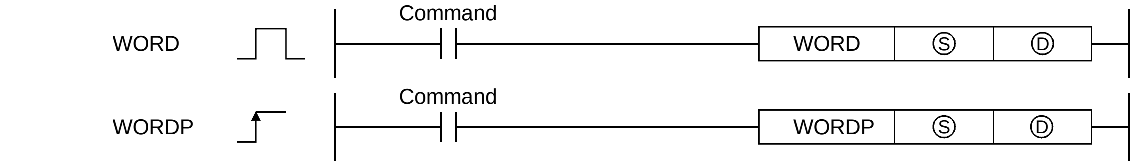


QCPU			QnA	Q4AR
PLC CPU		Process CPU		
Basic	High Performance			
○	○	○	○	○

6.3.6 Conversion from BIN 32-bit to BIN 16-bit data (WORD, WORDP)

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant K, H	Other
	Bit	Word		Bit	Word				
Ⓢ	○							○	–
Ⓓ	○							–	–

[Instruction Symbol] [Execution Condition]



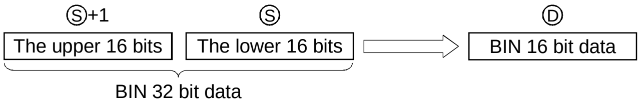
[Set Data]

Set Data	Meaning	Data Type
Ⓢ	Head number of device where BIN 32-bit data is stored	BIN 32 bits
Ⓓ	Head number of device where BIN 16-bit data will be stored after conversion	BIN 16 bits

[Functions]

Converts BIN 32-bit data at device designated by Ⓢ to BIN 16-bit data with sign, and stores the result at a device designated by Ⓓ.

Devices can be designated in the range from -32768 to 32767



[Operation Errors]

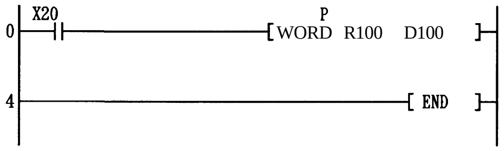
(1) In the following cases an operation error occurs, the error flag (SM0) turns ON, and an error code is stored at SD0.

- The contents of the data designated by Ⓢ+1 and Ⓢ are outside the -32768 and 32767 range. (Error code: 4100)

[Program Example]

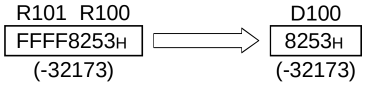
(1) The following program converts the BIN 32-bit data at R100 and R101 to BIN 16-bit data when X20 is ON, and stores it at D100.

[Ladder Mode]



[List Mode]

Steps	Instruction	Device
0	LD	X20
1	WORD P	R100 D100
4	END	

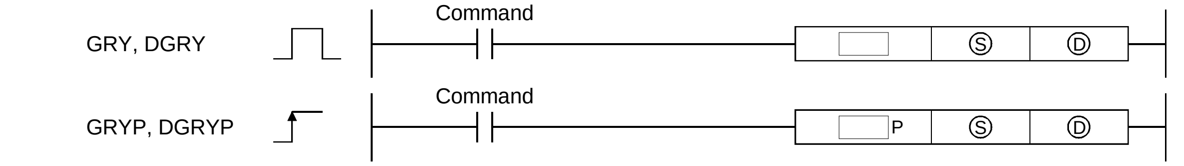


QCPU			QnA	Q4AR
PLC CPU		Process CPU		
Basic	High Performance			
○	○	○	○	○

6.3.7 Conversion from BIN 16 and 32-bit data to Gray code (GRY, GRYP, DGRY, DGRYP)

Set Data	Usable Devices									
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant K, H	Other
	Bit	Word		Bit	Word				
Ⓢ	○							○	–
Ⓓ	○							–	–

[Instruction Symbol] [Execution Condition] □ indicates GRY or DGRY



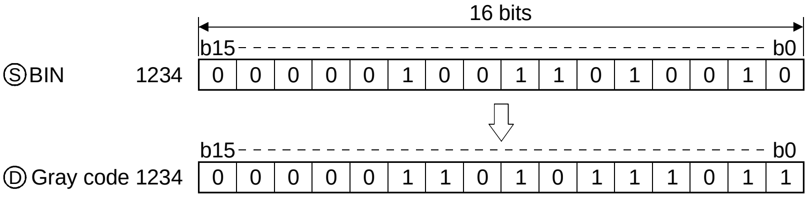
[Set Data]

Set Data	Meaning	Data Type
Ⓢ	BIN data, or head number of the device where BIN data is stored	BIN 16/32 bits
Ⓓ	Head number of device to store Gray code after conversion	Gray code

[Functions]

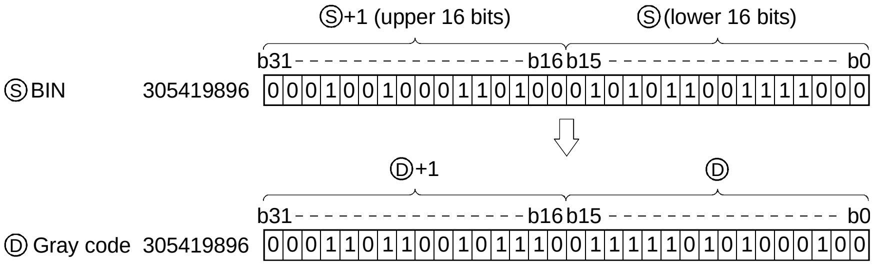
GRY

Converts BIN 16-bit data at the device designated by Ⓢ to Gray code, and stores result at device designated by Ⓓ.



DGRY

Converts BIN 32-bit data at the device designated by Ⓢ to Gray code, and stores result at device designated by Ⓓ.



[Operation Errors]

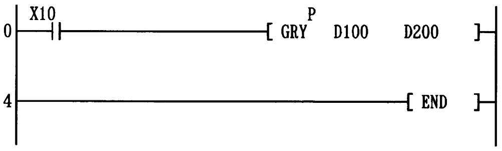
(1) In the following cases an operation error occurs, the error flag (SM0) turns ON, and an error code is stored at SD0.

- The data at Ⓢ is a negative number.

[Program Example]

(1) The following program converts the BIN data at D100 to Gray code when X10 is ON, and stores result at D200.

[Ladder Mode]

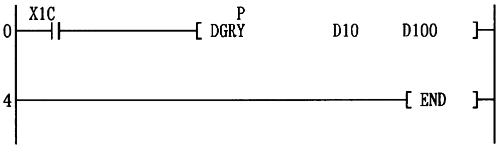


[List Mode]

Steps	Instruction	Device
0	LD	X10
1	GRYP	D100 D200
4	END	

(2) The following program converts the BIN data at D10 and D11 to Gray code when X1C is ON, and stores it at D100 and D101.

[Ladder Mode]



[List Mode]

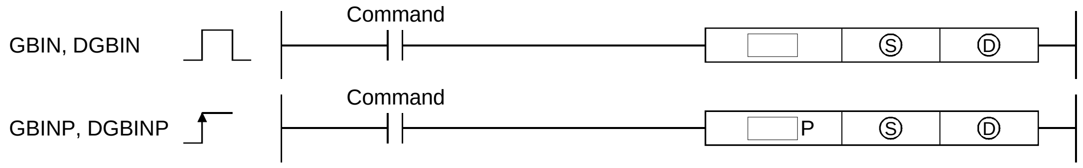
Steps	Instruction	Device
0	LD	X1C
1	DGRYP	D10 D100
4	END	

QCPU			QnA	Q4AR
PLC CPU		Process CPU		
Basic	High Performance			
○	○	○	○	○

6.3.8 Conversion of Gray code to BIN 16 and 32-bit data (GBIN, GBINP, DGBIN, DGBINP)

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant K, H	Other
	Bit	Word		Bit	Word				
Ⓢ	○							○	–
Ⓓ	○							–	–

[Instruction Symbol] [Execution Condition] □ indicates GBIN or DGBIN



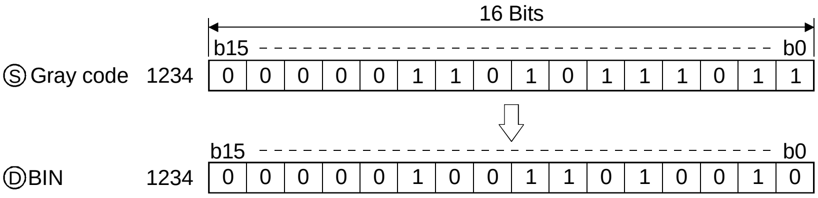
[Set Data]

Set Data	Meaning	Data Type
Ⓢ	Gray code data or the head number of device where Gray code data is being stored	Gray code
Ⓓ	Head number of the device to store BIN data after conversion	BIN 16/32 bits

[Functions]

GBIN

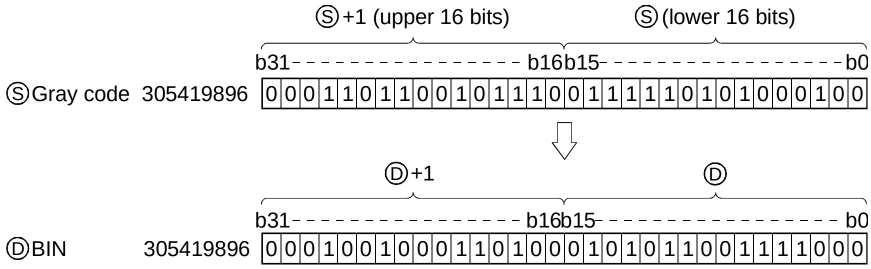
Converts Gray code data at device designated by Ⓢ to BIN 16-bit data and stores at device designated by Ⓓ.



		b15															b0
Ⓢ Gray code	1234	0	0	0	0	0	1	1	0	1	0	1	1	1	0	1	1
Ⓓ BIN	1234	0	0	0	0	0	1	0	0	1	1	0	1	0	0	1	0

DGBIN

Converts Gray code data at device designated by Ⓢ to BIN 32-bit data and stores at device designated by Ⓓ.



		b31															b16	b15															b0
Ⓢ Gray code	305419896	0	0	0	1	1	0	1	1	0	0	1	0	1	1	1	0	0	1	1	1	1	1	0	1	0	1	0	0	0	1	0	0
Ⓓ BIN	305419896	0	0	0	1	0	0	1	0	0	0	1	1	0	1	0	0	0	1	0	1	0	1	1	0	0	1	1	1	1	0	0	0

[Operation Errors]

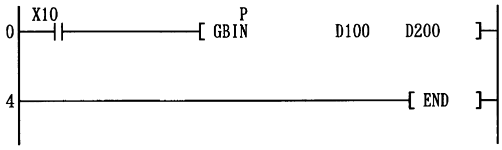
(1) In the following cases an operation error occurs, the error flag (SM0) turns ON, and an error code is stored at SD0.

- Data at Ⓢ when GBIN instruction was issued is outside the 0 to 32767 range.
- Data at Ⓢ when DGBIN instruction was issued is outside the 0 to 2147483647 range.

[Program Example]

(1) The following program converts the Gray code data at D100 when X10 is ON to BIN data, and stores the result at D200.

[Ladder Mode]

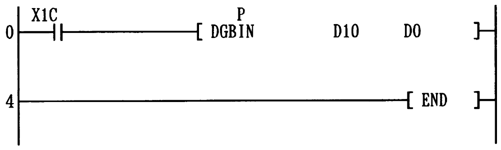


[List Mode]

Steps	Instruction	Device
0	LD	X10
1	GBINP	D100 D200
4	END	

(2) The following program converts the Gray code data at D10 and D11 to BIN data when X1C is ON, and stores the result at D0 and D1.

[Ladder Mode]



[List Mode]

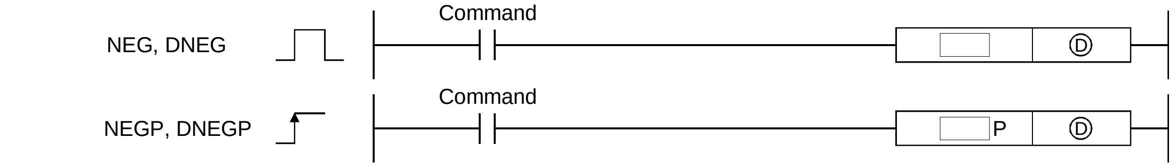
Steps	Instruction	Device
0	LD	X1C
1	DGBINP	D10 D0
4	END	

QCPU			QnA	Q4AR
PLC CPU		Process CPU		
Basic	High Performance			
○	○	○	○	○

6.3.9 Complement of 2 of BIN 16- and 32-bit data (sign reversal) (NEG, NEGP, DNEG, DNEGP)

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant K, H	Other
	Bit	Word		Bit	Word				
Ⓓ	○							—	

[Instruction Symbol] [Execution Condition] □ indicates NEG or DNEG



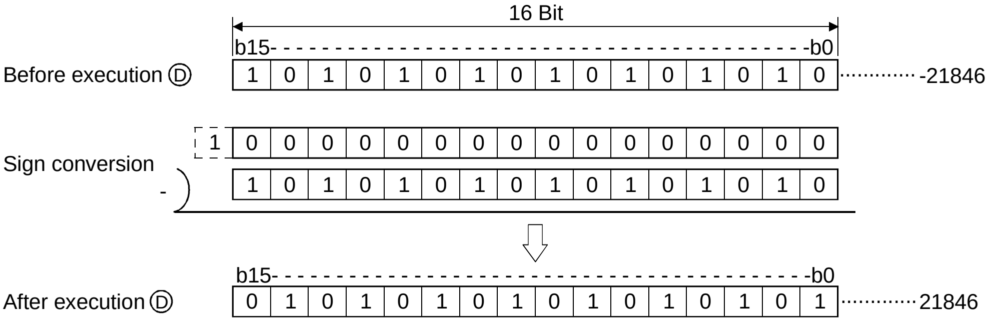
[Set Data]

Set Data	Meaning	Data Type
Ⓓ	Head number of device storing data for the complement of 2 operation	BIN 16/32 bits

[Functions]

NEG

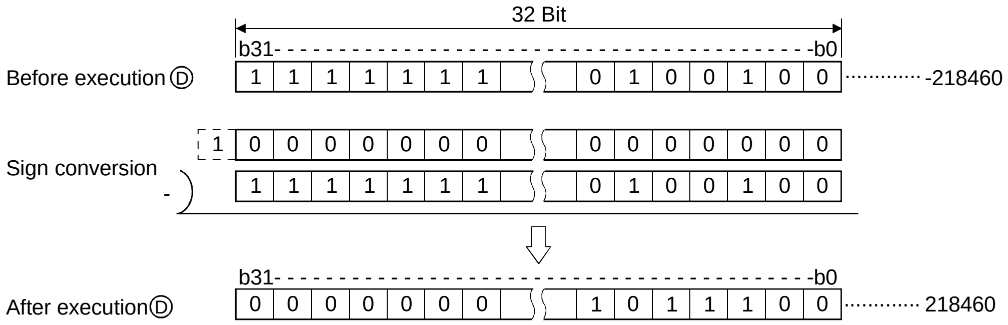
(1) Reverses the sign of the 16-bit device designated by Ⓓ and stores at the device designated by Ⓓ.



(2) Used when reversing positive and negative signs.

DNEG

(1) Reverses the sign of the 32-bit device designated by Ⓓ and stores at the device designated by Ⓓ.



(2) Used when reversing positive and negative signs.

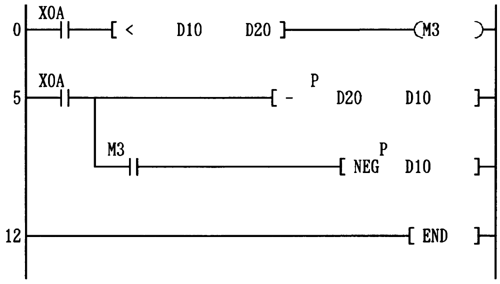
[Operation Errors]

(1) There are no operation errors associated with the NEG(P) or DNEG(P) instructions.

[Program Example]

(1) The following program calculates a total for the data at D10 through D20 when XA goes ON, and seeks an absolute value if the result is negative.

[Ladder Mode]



M3 goes ON if D10 is smaller than D20

Subtracts D20 from D10

Seeks an absolute value (complement of 2) when M3 is ON

[List Mode]

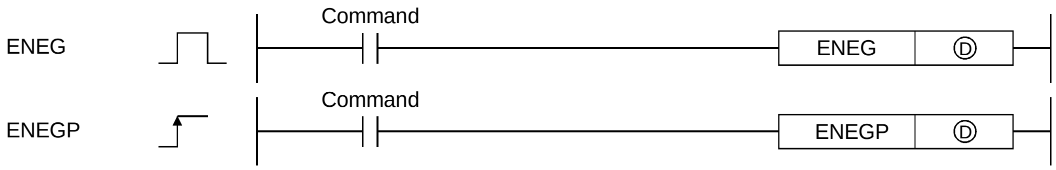
Steps	Instruction	Device
0	LD	X0A
1	AND<	D10
		D20
4	OUT	M3
5	LD	X0A
6	-P	D20
		D10
8	AND	M3
9	NEGP	D10
12	END	

QCPU			QnA	Q4AR
PLC CPU		Process CPU		
Basic	High Performance			
×	○	○	○	○

6.3.10 Sign reversal for floating decimal point data (ENEG, ENEGP)

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant K, H	Other
	Bit	Word		Bit	Word				
Ⓓ	–	○		–	○		–	–	

[Instruction Symbol] [Execution Condition]



[Set Data]

Set Data	Meaning	Data Type
Ⓓ	Head number of device storing floating decimal point data for which sign will be inverted	Real number

[Functions]

(1) Reverses the sign of the floating decimal point type real number data designated by Ⓓ, and stores at the device designated by Ⓓ.

(2) Used when reversing positive and negative signs.

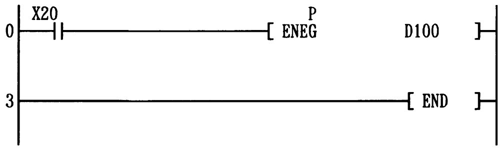
[Operation Errors]

(1) There are no errors associated with the ENEG(P) instruction.

[Program Example]

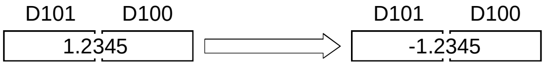
(1) The following program inverts the sign of the floating decimal point type real number data at D100 and D101 when X20 goes ON, and stores result at D100 and D101.

[Ladder Mode]



[List Mode]

Steps	Instruction	Device
0	LD	X20
1	ENEGP	D100
3	END	

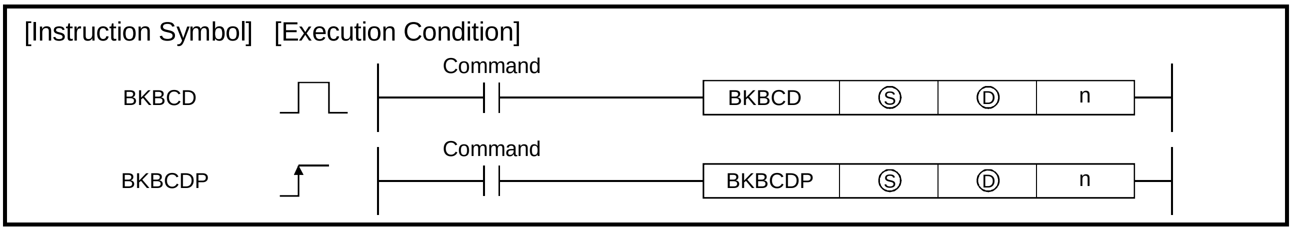


QCPU			QnA	Q4AR
PLC CPU		Process CPU		
Basic	High Performance			
○	○	○	○	○

6.3.11 Conversion from block BIN 16-bit data to BCD 4-digit data (BKBCD, BKBCDP)

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant K, H	Other
	Bit	Word		Bit	Word				
Ⓢ	–	○		–					–
Ⓓ	–	○		–					–
n	○	○		○					–

[Instruction Symbol] [Execution Condition]

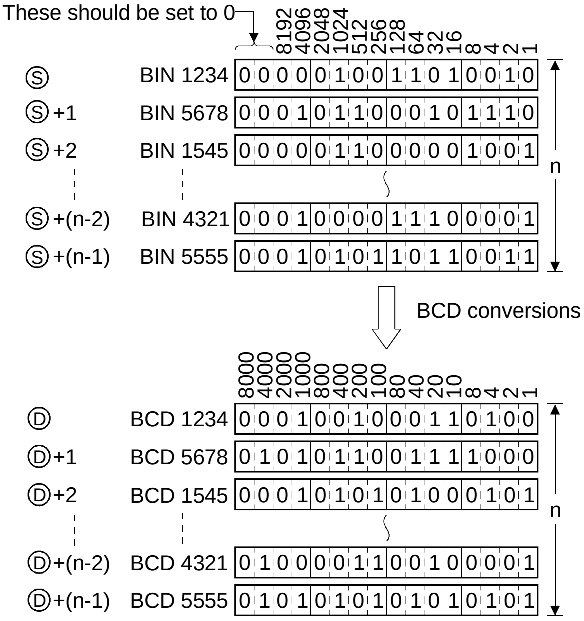


[Set Data]

Set Data	Meaning	Data Type
Ⓢ	Head number of device storing BIN data	BIN 16 bits
Ⓓ	Head number of device which will store BCD data after conversion	
n	Number of data blocks converted	

[Functions]

(1) Converts BIN data (0 to 9999) n-points from device designated by Ⓢ to BCD, and stores result following the device designated by Ⓓ.



Ⓢ BIN 1234 0000 0100 1101 0010
Ⓢ+1 BIN 5678 0001 0110 0010 1110
Ⓢ+2 BIN 1545 0000 0110 0000 1001
Ⓢ+(n-2) BIN 4321 0001 0000 1110 0001
Ⓢ+(n-1) BIN 5555 0001 0101 1011 0011

Ⓓ BCD 1234 0001 0010 0011 0100
Ⓓ+1 BCD 5678 0101 0110 0111 1000
Ⓓ+2 BCD 1545 0001 0101 0100 0101
Ⓓ+(n-2) BCD 4321 0100 0011 0010 0001
Ⓓ+(n-1) BCD 5555 0101 0101 0101 0101

[Operation Errors]

(1) In the following cases an operation error occurs, the error flag (SM0) turns ON, and an error code is stored at SD0.

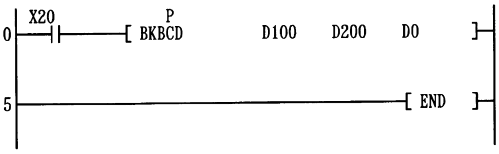
- The range n-points from the device at Ⓢ or Ⓓ exceeds the relevant device. (Error code: 4101)
- The data n-points from the device designated by Ⓢ is outside the 0 to 9999 range. (Error code: 4100)
- The Ⓢ and Ⓓ devices overlap. (Error code: 4101)

(2) See Section 3.6 for information regarding errors during index modification.

[Program Example]

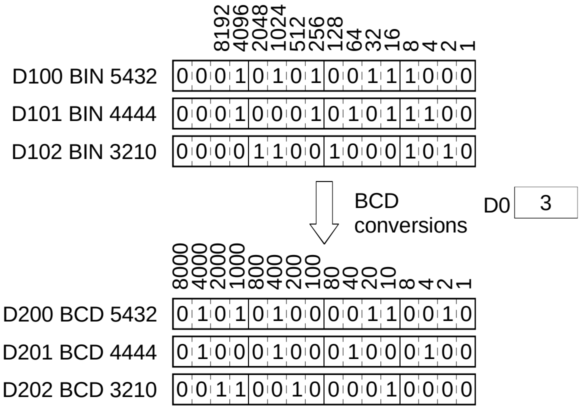
(1) The following program converts BIN 16-bit data the number of points from D100 corresponding to the value stored at D0 to BCD when X20 goes ON, and stores the result following D200.

[Ladder Mode]



[List Mode]

Steps	Instruction	Device
0	LD	X20
1	BKBCDP	D100 D200 D0
5	END	



	32768	16384	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1
D100 BIN 5432	0	0	0	1	0	1	0	1	0	0	1	1	1	0	0	0
D101 BIN 4444	0	0	0	1	0	0	0	1	0	1	0	1	1	1	0	0
D102 BIN 3210	0	0	0	0	1	1	0	0	1	0	0	0	1	0	1	0

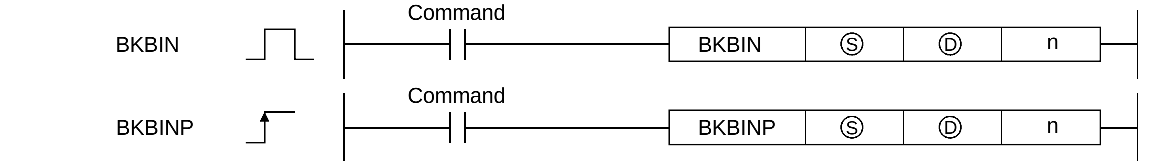
	8000	4000	2000	1000	800	400	200	100	80	40	20	10	8	4	2	1
D200 BCD 5432	0	1	0	1	0	1	0	0	0	0	1	1	0	0	1	0
D201 BCD 4444	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0
D202 BCD 3210	0	0	1	1	0	0	1	0	0	0	0	1	0	0	0	0

QCPU			QnA	Q4AR
PLC CPU		Process CPU		
Basic	High Performance			
○	○	○	○	○

6.3.12 Conversion from block BCD 4-digit data to block BIN 16-bit data (BKBIN, BKBINP)

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant K, H	Other
	Bit	Word		Bit	Word				
Ⓢ	–	○		–					–
Ⓓ	–	○		–					–
n	○	○		○					–

[Instruction Symbol] [Execution Condition]

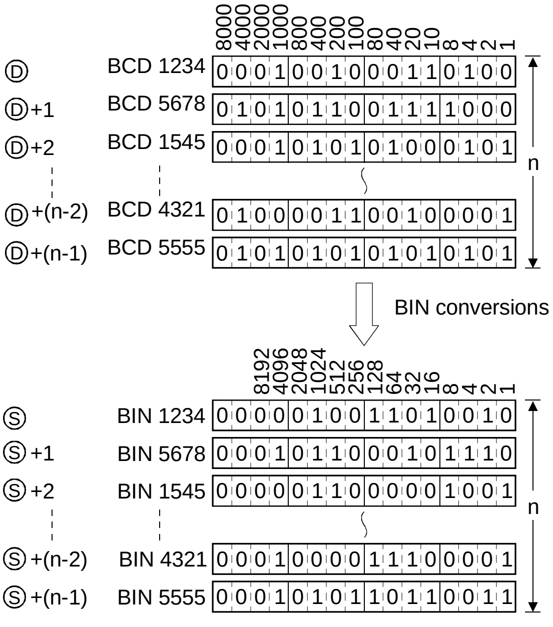


[Set Data]

Set Data	Meaning	Data Type
Ⓢ	Head number of device storing BCD data	BIN 16 bits
Ⓓ	Head number of the device to store BIN data after conversion	
n	Number of data blocks converted	

[Functions]

(1) Converts BCD data (0 to 9999) n-points from device designated by Ⓢ to BIN, and stores result following the device designated by Ⓓ.



		8000	4000	2000	1000	800	400	200	100	80	40	20	10	8	4	2	1
Ⓓ	BCD 1234	0	0	0	1	0	0	1	0	0	0	1	1	0	1	0	0
Ⓓ+1	BCD 5678	0	1	0	1	0	1	1	0	0	1	1	1	1	0	0	0
Ⓓ+2	BCD 1545	0	0	0	1	0	1	0	1	0	1	0	0	0	1	0	1
Ⓓ+(n-2)	BCD 4321	0	1	0	0	0	0	1	1	0	0	1	0	0	0	0	1
Ⓓ+(n-1)	BCD 5555	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

				8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1
Ⓢ	BIN 1234	0	0	0	0	0	1	0	0	1	1	0	1	0	0	1	0
Ⓢ+1	BIN 5678	0	0	0	1	0	1	1	0	0	0	1	0	1	1	1	0
Ⓢ+2	BIN 1545	0	0	0	0	0	1	1	0	0	0	0	0	1	0	0	1
Ⓢ+(n-2)	BIN 4321	0	0	0	1	0	0	0	0	1	1	1	0	0	0	0	1
Ⓢ+(n-1)	BIN 5555	0	0	0	1	0	1	0	1	1	0	1	1	0	0	1	1

[Operation Errors]

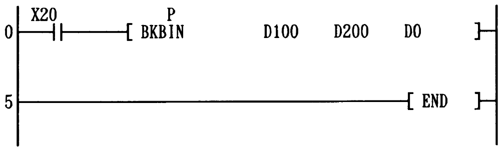
(1) In the following cases an operation error occurs, the error flag (SM0) turns ON, and an error code is stored at SD0.

- The range n-points from the Ⓢ or Ⓓ device exceeds the relevant device.
- The data n-points at the Ⓢ device is outside the 0 to 9999 range.
- The Ⓢ and Ⓓ devices overlap.

[Program Example]

(1) The following program converts BCD data the number of points from D100 corresponding to the value stored at D0 to BIN data when X20 goes ON, and stores the result from D200 onward.

[Ladder Mode]



[List Mode]

Steps	Instruction	Device
0	LD	X20
1	BKBINP	D100 D200 D0
5	END	

	8000	4000	2000	1000	800	400	200	100	80	40	20	10	8	4	2	1
D100 BCD 8080	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
D101 BCD 7654	0	1	1	1	0	1	1	0	0	1	0	1	0	1	0	0
D102 BCD 9999	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1

BIN conversions (when D0=3)

			8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1
D200 BIN 8080	0	0	0	1	1	1	1	1	1	0	0	1	0	0	0	0
D201 BIN 7654	0	0	0	1	1	1	0	1	1	1	1	0	0	1	1	0
D202 BIN 9999	0	0	1	0	0	1	1	1	0	0	0	0	1	1	1	1

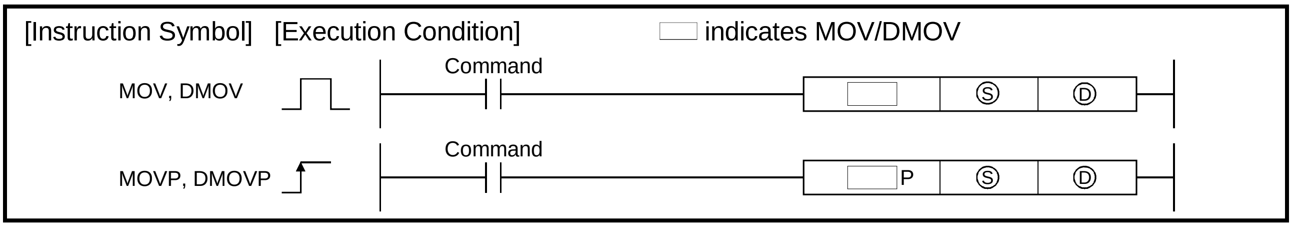
QCPU			QnA	Q4AR
PLC CPU		Process CPU		
Basic	High Performance			
○	○	○	○	○

6.4 Data Transfer Instructions

6.4.1 16-bit and 32-bit data transfers (MOV, MOVP, DMOV, DMOVP)

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant K, H	Other
	Bit	Word		Bit	Word				
Ⓢ	○							○	–
Ⓓ	○							–	–

[Instruction Symbol] [Execution Condition] □ indicates MOV/DMOV



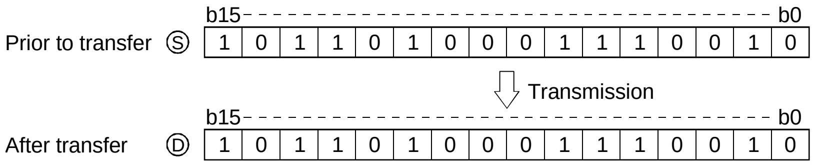
[Set Data]

Set Data	Meaning	Data Type
Ⓢ	Transfer data, or number of device storing transfer data	BIN 16/32 bits
Ⓓ	Number of device to store transferred data	

[Functions]

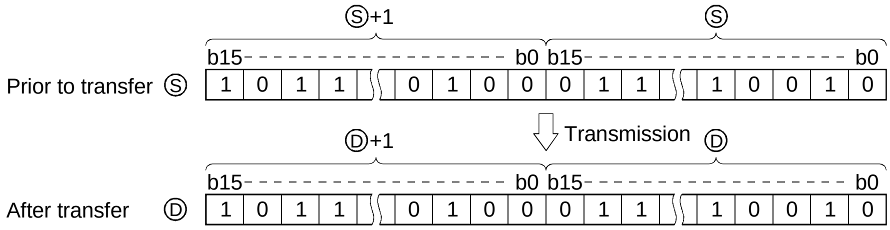
MOV

(1) Transfers the 16-bit data from the device designated by Ⓢ to the device designated by Ⓓ.



DMOV

(2) Transfers 32-bit data at the device designated by Ⓢ to the device designated by Ⓓ.



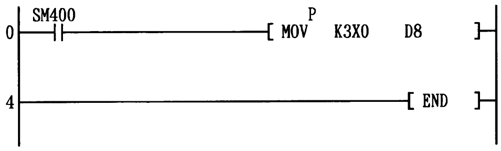
[Operation Errors]

(1) There are no operation errors associated with the MOV(P) or DMOV(P) instructions.

[Program Example]

(1) The following program stores input data from X0 to XB at D8.

[Ladder Mode]

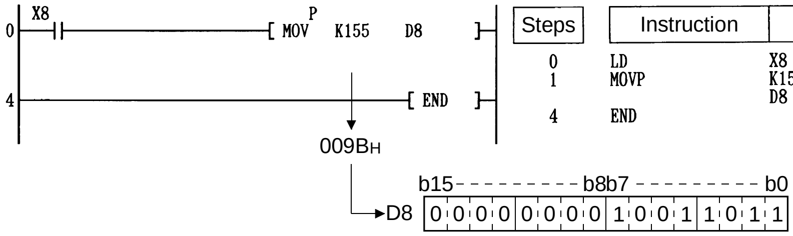


[List Mode]

Steps	Instruction	Device
0	LD	SM400
1	MOVP	K3X0 D8
4	END	

(2) The following program stores the constant K155 at D8 when X8 goes ON.

[Ladder Mode]

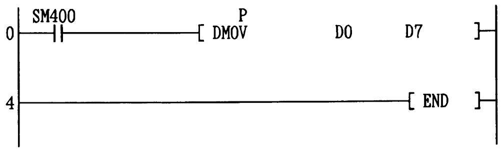


[List Mode]

Steps	Instruction	Device
0	LD	X8
1	MOVP	K155 D8
4	END	

(3) The following program stores the data from D0 and D1 at D7 and D8.

[Ladder Mode]

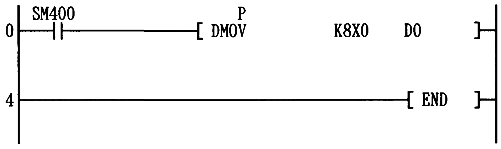


[List Mode]

Steps	Instruction	Device
0	LD	SM400
1	DMOVP	D0 D7
4	END	

(4) The following program stores the data from X0 to X1F at D0 and D1.

[Ladder Mode]



[List Mode]

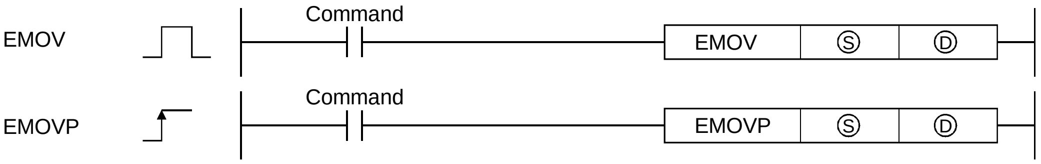
Steps	Instruction	Device
0	LD	SM400
1	DMOVP	K8X0 D0
4	END	

QCPU			QnA	Q4AR
PLC CPU		Process CPU		
Basic	High Performance			
×	○	○	○	○

6.4.2 Floating decimal point data transfers (EMOV, EMOVP)

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant E	Other
	Bit	Word		Bit	Word				
Ⓢ	–	○		–	○		–	○	–
Ⓓ	–	○		–	○		–	–	–

[Instruction Symbol] [Execution Condition]

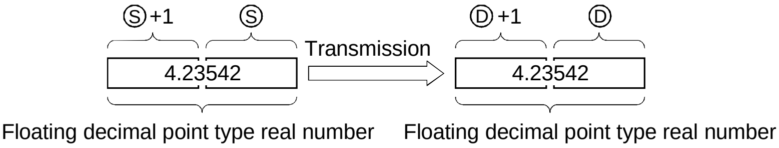


[Set Data]

Set Data	Meaning	Data Type
Ⓢ	Transfer data, or number of device storing transfer data	Real number
Ⓓ	Number of device to store transferred data	

[Functions]

(1) Transfers floating decimal point type real number data being stored at the device designated by Ⓢ to a device designated by Ⓓ.


Ⓢ+1
Ⓢ
Transmission
Ⓓ+1
Ⓓ
4.23542
4.23542
Floating decimal point type real number
Floating decimal point type real number

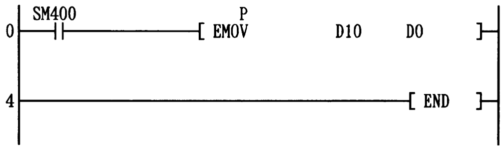
[Operation Errors]

(1) There are no operation errors associated with the EMOV(P) instruction.

[Program Example]

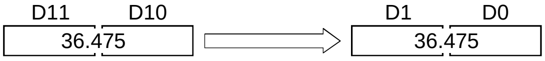
(1) The following program stores the real numbers at D10 and D11 at D0 and D1.

[Ladder Mode]



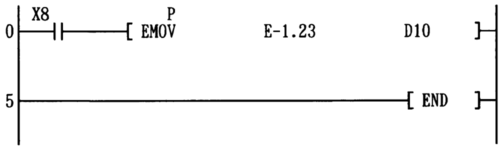
[List Mode]

Steps	Instruction	Device
0	LD	SM400
1	EMOVP	D10 D0
4	END	



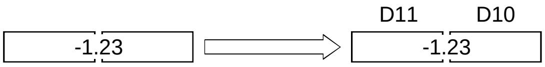
(2) The following program stores the real number -1.23 at D10 and D11 when X8 is ON.

[Ladder Mode]



[List Mode]

Steps	Instruction	Device
0	LD	X8
1	EMOVP	E-1.23 D10
5	END	

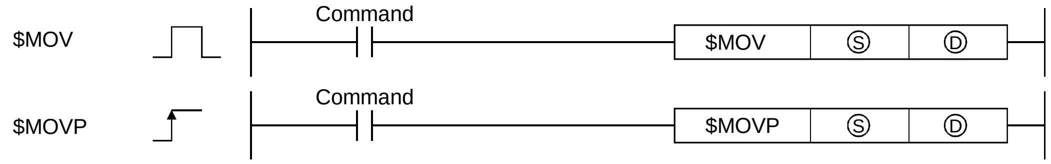


QCPU			QnA	Q4AR
PLC CPU		Process CPU		
Basic	High Performance			
○	○	○	○	○

6.4.3 Character string transfers ($MOV, $MOVP)

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant $	Other
	Bit	Word		Bit	Word				
Ⓢ	—	○		—				○	—
Ⓓ	—	○		—				—	—

[Instruction Symbol] [Execution Condition]

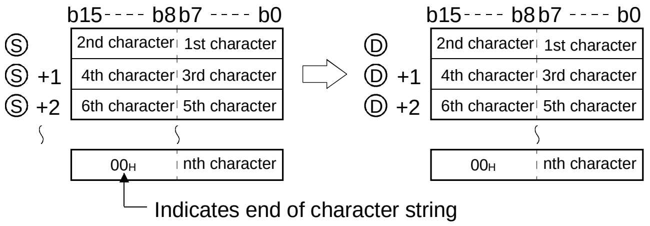


[Set Data]

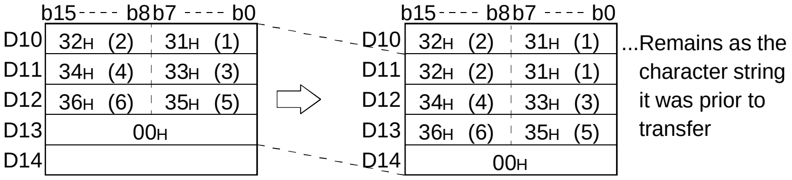
Set Data	Meaning	Data Type
Ⓢ	Character string to be transferred (maximum number of characters in a string: 16 characters for QnA/Q4AR, 32 characters for QCPU), or the head number of the device storing character string.	Character string
Ⓓ	Head number of device to store transferred character string	

[Functions]

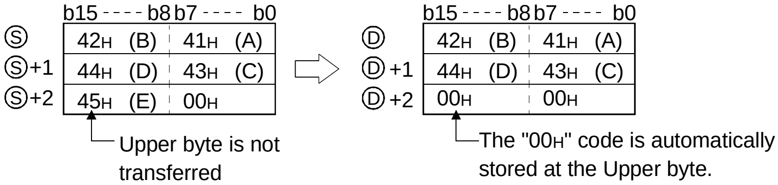
(1) Transfers character string data stored from device number designated by Ⓢ from device number designated by Ⓓ onward.
A character string transfer involves the transfer of data from the device number designated by Ⓢ to the device number storing the "00H" code in one operation.



(2) Processing will be performed without error even in cases where the range for the devices storing the character data to be transferred (Ⓢ to Ⓢ+n) overlaps with the range of the devices which will store the character string data after it has been transferred (Ⓓ to Ⓓ+n).
The following occurs when the character string data that had been stored from D10 to D13 is transferred to D11 to D14:



(3) If the "00H" code is being stored at lower bytes of Ⓢ+n, "00H" will be stored at both the higher bytes and the lower bytes of Ⓓ+n.



[Operation Errors]

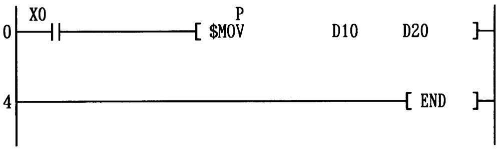
(1) In the following cases an operation error occurs, the error flag (SM0) turns ON, and an error code is stored at SD0.

- There is no "00H" code stored between the device number designated by Ⓢ and the relevant device. (Error code: 4101)
- It is not possible to store the entire designated character string in the number of points from the device designated by Ⓓ to the final device number cited. (Error code: 4101)

[Program Example]

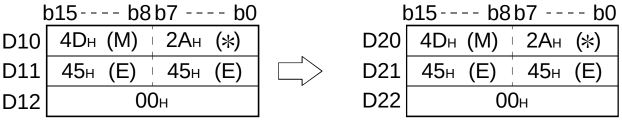
(1) The character string data stored in D10 to D12 is transfered to D20 to D22 when X0 goes ON.

[Ladder Mode]



[List Mode]

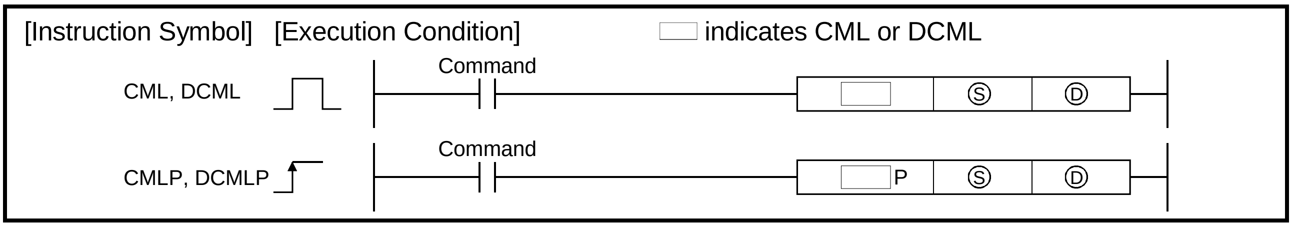
Steps	Instruction	Device
0	LD	X0
1	$MOVP	D10 D20
4	END	



QCPU			QnA	Q4AR
PLC CPU		Process CPU		
Basic	High Performance			
○	○	○	○	○

6.4.4 16-bit and 32-bit negation transfers (CML, CMLP, DCML, DCMLP)

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant K, H	Other
	Bit	Word		Bit	Word				
Ⓢ	○							○	–
Ⓓ	○							–	–



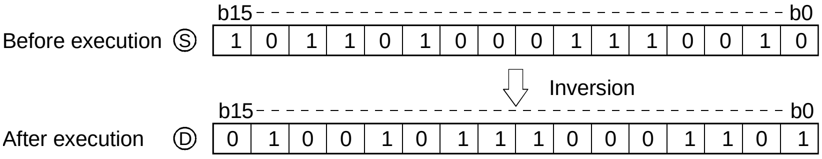
[Set Data]

Set Data	Meaning	Data Type
Ⓢ	Data to be inverted, or number of device storing this data	BIN 16/32 bits
Ⓓ	Number of device that will store results of inversion	

[Functions]

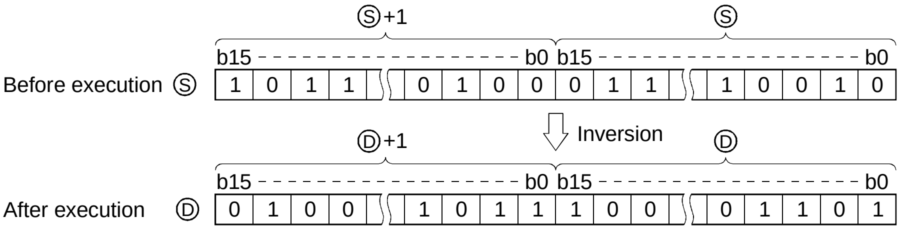
CML

(1) Inverts 16-bit data designated by Ⓢ bit by bit, and transfers the result to the device designated by Ⓓ.



DCML

(1) Inverts 32-bit data designated by Ⓢ bit by bit, and transfers the result to the device designated by Ⓓ.



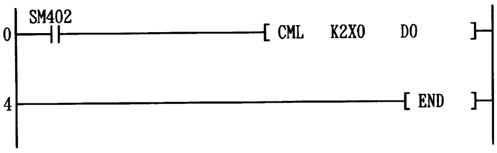
[Operation Errors]

(1) There are no operation errors associated with the CML(P) or DCML(P) instructions.

[Program Example]

(1) The following program inverts the data from X0 to X7, and transfers result to D0.

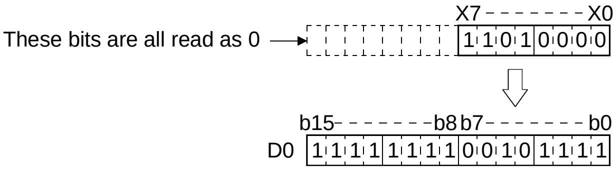
[Ladder Mode]



[List Mode]

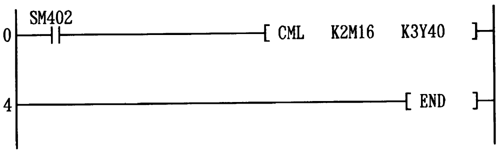
Steps	Instruction	Device
0	LD	SM402
1	CML	K2X0 D0
4	END	

When the number of bits at Ⓢ is less than the number of bits at Ⓓ


X7 - - - - - - - X0
These bits are all read as 0
1 1 0 1 0 0 0 0
b15 - - - - - - - b8 b7 - - - - - - - b0
D0
1 1 1 1 1 1 1 1 0 0 1 0 1 1 1 1

(2) The following program inverts the data at M16 to M23, and transfers the result to Y40 to Y47.

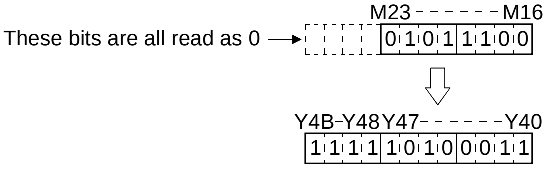
[Ladder Mode]



[List Mode]

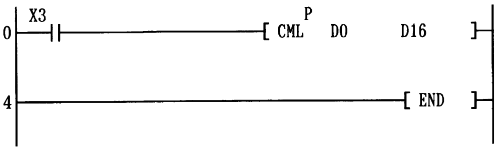
Steps	Instruction	Device
0	LD	SM402
1	CML	K2M16 K3Y40
4	END	

When the number of bits at Ⓢ is less than the number of bits at Ⓓ



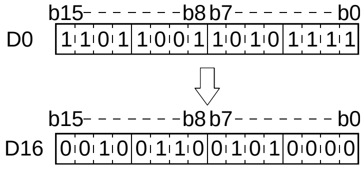
(3) The following program inverts the data at D0 when X3 is ON, and stores the result at D16.

[Ladder Mode]



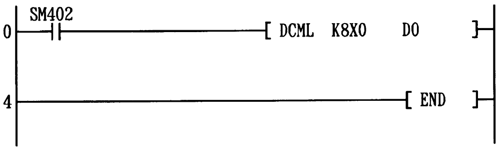
[List Mode]

Steps	Instruction	Device
0	LD	X3
1	CMLP	D0 D16
4	END	


b15 - - - - - - b8 b7 - - - - - - - b0
D0
1 1 0 1 1 0 0 1 1 0 1 0 1 1 1 1
b15 - - - - - - b8 b7 - - - - - - - b0
D16
0 0 1 0 0 1 1 0 0 1 0 1 0 0 0 0

(4) The following program inverts the data at X0 to X1F, and transfers results to D0 and D1.

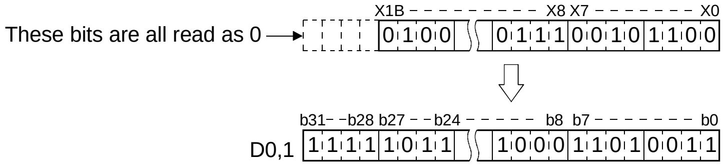
[Ladder Mode]



[List Mode]

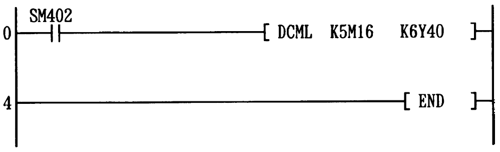
Steps	Instruction	Device
0	LD	SM402
1	DCML	K8X0 D0
4	END	

When the number of bits at Ⓢ is less than the number of bits at Ⓓ



(5) The following program inverts the data at M16 to M35, and transfers it to Y40 to Y63.

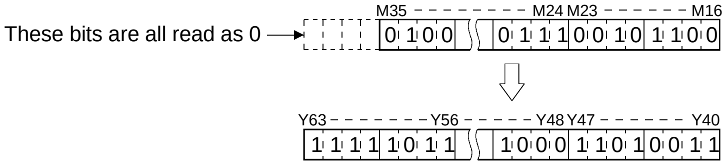
[Ladder Mode]



[List Mode]

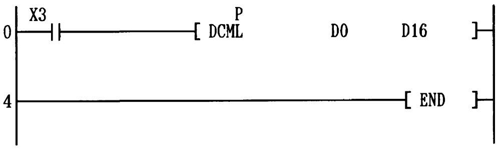
Steps	Instruction	Device
0	LD	SM402
1	DCML	K5M16 K6Y40
4	END	

When the number of bits at Ⓢ is less than the number of bits at Ⓓ



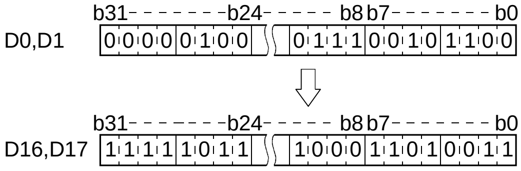
(6) Inverts the data at D0 and D1 when X3 is ON, and stores the result at D16 and D17.

[Ladder Mode]



[List Mode]

Steps	Instruction	Device
0	LD	X3
1	DCMLP	D0 D16
4	END	

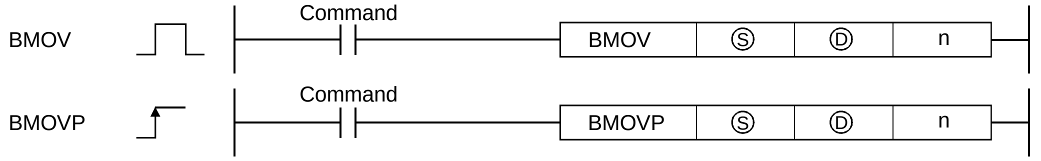


QCPU				QnA	Q4AR
PLC CPU		Process CPU			
Basic	High Performance				
○	○	○		○	○

6.4.5 Block 16-bit data transfers (BMOV, BMOVP)

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant K, H	Other
	Bit	Word		Bit	Word				
Ⓢ	○						—		—
Ⓓ	○						—		—
n	○						○		—

[Instruction Symbol] [Execution Condition]

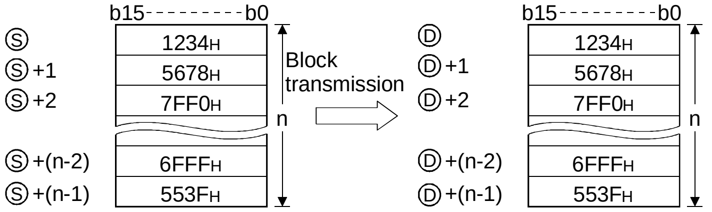


[Set Data]

Set Data	Meaning	Data Type
Ⓢ	Head number of device storing data to be transferred	BIN 16 bits
Ⓓ	Head number of destination device	
n	Number of transfers (If special direct device (U□\G□) is used: 1 to 6144 (QnACPU))	

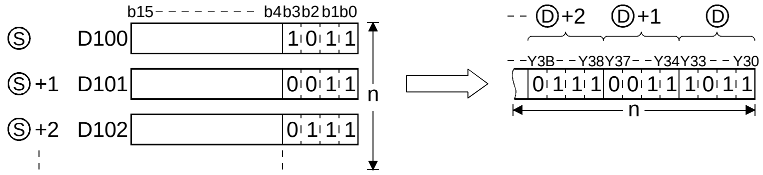
[Functions]

(1) Transfers in batch 16-bit data n-points from the device designated by Ⓢ to location n-points from the device designated by Ⓓ.



(2) Transfers can be accomplished even in cases where there is an overlap between the source and destination device.
In the case of transmission to the smaller device number, transmission is from Ⓢ; for transmission to the larger device number, transmission is from Ⓢ + (n-1).

(3) When Ⓢ is a word device and Ⓓ is a bit device, the object for the word device will be the number of bits designated by the bit device digit designation.
If K1Y30 has been designated by Ⓓ, the lower four bits of the word device designated by Ⓢ will become the object.



(4) If bit device has been designated for Ⓢ and Ⓓ, then Ⓢ and Ⓓ should always have the same number of digits.

(5) Only either of Ⓢ or Ⓓ can be designated for the MELSECNET/10(H) direct device and intelligent function module/special function module device.

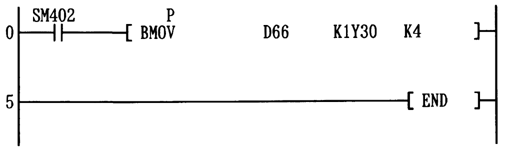
[Operation Errors]

(1) In the following cases an operation error occurs, the error flag (SM0) turns ON, and an error code is stored at SD0.
- The device range n-points from Ⓢ or Ⓓ exceeds the relevant device. (Error code: 4101)
- The number of transfers exceeds 6144 when a special direct device is used. (QnACPU) (Error code: 4101)

[Program Example]

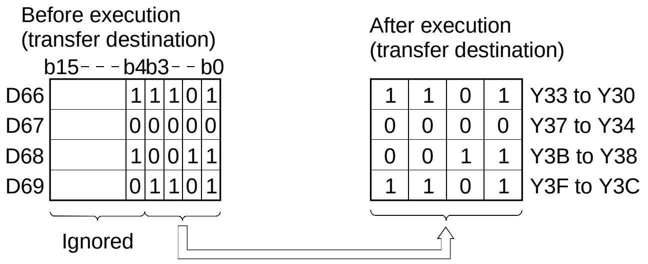
(1) The following program outputs the lower 4 bits of data at D66 to D69 to Y30 to Y3F in 4-point units.

[Ladder Mode]



[List Mode]

Steps	Instruction	Device
0	LD	SM402
1	BMOVP	D66 K1Y30 K4
5	END	



Before execution (transfer destination)

	b15- - -	b4	b3	-	-	b0
D66		1	1	1	0	1
D67		0	0	0	0	0
D68		1	0	0	1	1
D69		0	1	1	0	1

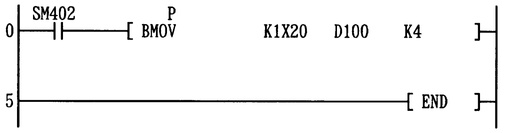
Ignored

After execution (transfer destination)

1	1	0	1	Y33 to Y30
0	0	0	0	Y37 to Y34
0	0	1	1	Y3B to Y38
1	1	0	1	Y3F to Y3C

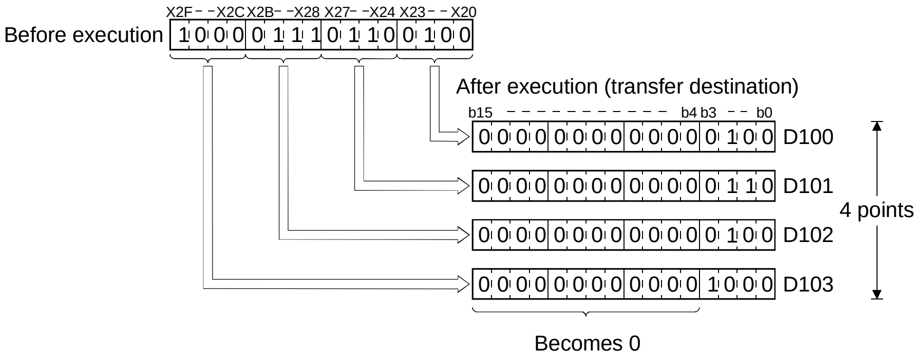
(2) The following program outputs the data at X20 to X2F to D100 to D103 in 4-point units.

[Ladder Mode]



[List Mode]

Steps	Instruction	Device
0	LD	SM402
1	BMOVP	K1X20 D100 K4
5	END	



	X2F- -X2C	X2B- -X28	X27- -X24	X23- -X20
Before execution	1 0 0 0	0 1 1 1	0 1 1 0	0 1 0 0

After execution (transfer destination)

	b15 - - -		b4	b3 - - b0	
0 0 0 0	0 0 0 0	0 0 0 0	0 1 0 0	D100
0 0 0 0	0 0 0 0	0 0 0 0	0 1 1 0	D101
0 0 0 0	0 0 0 0	0 0 0 0	0 1 0 0	D102
0 0 0 0	0 0 0 0	0 0 0 0	1 0 0 0	D103

4 points

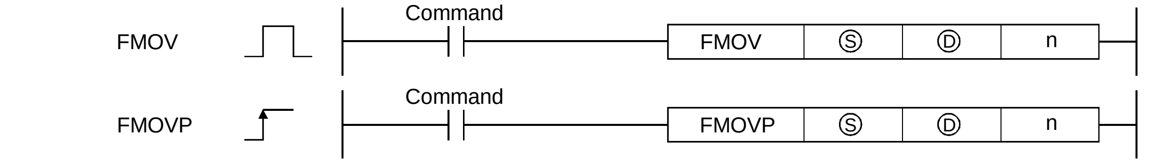
Becomes 0

QCPU			QnA	Q4AR
PLC CPU		Process CPU		
Basic	High Performance			
○	○	○	○	○

6.4.6 Identical 16-bit data block transfers (FMOV, FMOVP)

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant K, H	Other
	Bit	Word		Bit	Word				
Ⓢ	○						○		—
Ⓓ	○						—		—
n	○						○		—

[Instruction Symbol] [Execution Condition]

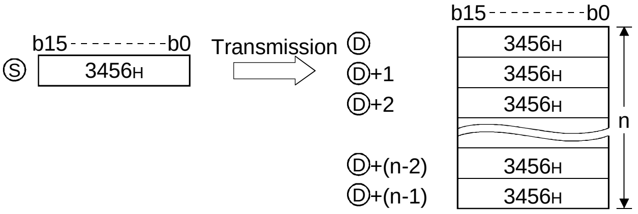


[Set Data]

Set Data	Meaning	Data Type
Ⓢ	Data to transfer, or head number of device storing data to transfer	BIN 16 bits
Ⓓ	Head number of destination device	
n	Number of transfers (If special direct device (U□\G□) is used: 1 to 6144 (QnACPU))	

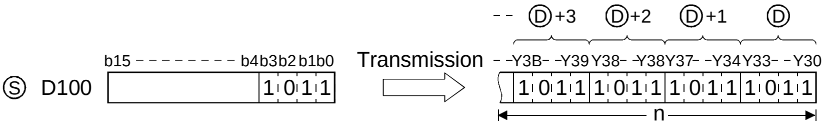
[Functions]

(1) Transfers 16-bit data from device designated by Ⓢ to location n-points from device designated by Ⓓ.



(2) In cases where Ⓢ designates a word device and Ⓓ a bit device, the number of bits designated by digit designation for the bit device will be the object bits for the word device.

If K1Y30 has been designated by Ⓓ, the object bits for the word device designated by Ⓢ will be the lower 4 bits.



(3) If bit device has been designated for Ⓢ and Ⓓ, then Ⓢ and Ⓓ should always have the same number of digits.

[Operation Errors]

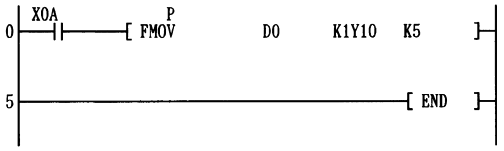
(1) In the following cases an operation error occurs, the error flag (SM0) turns ON, and an error code is stored at SD0.

- The device range n-points from Ⓓ exceeds the device range. (Error code: 4101)
- The number of transfers exceeds 6144 when a special direct device is used. (QnACPU) (Error code: 4101)

[Program Example]

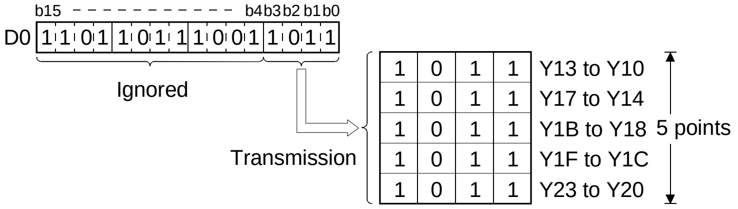
(1) The following program outputs the lower 4 bits of D0 when XA goes ON to Y10 to Y23 in 4-bit units.

[Ladder Mode]



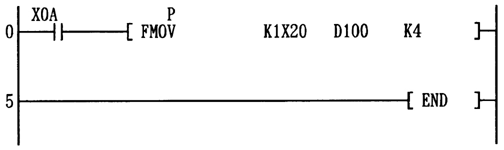
[List Mode]

Steps	Instruction	Device
0	LD	X0A
1	FMOVP	D0 K1Y10 K5
5	END	



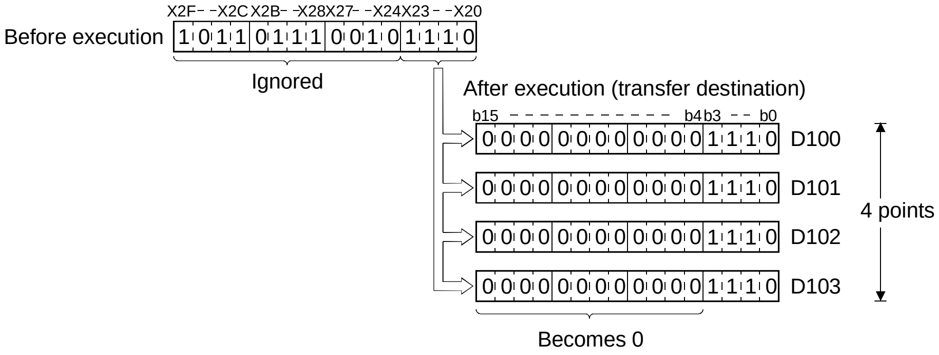
(2) The following program outputs the data at X20 through X23 to D100 through D103 when XA goes ON.

[Ladder Mode]



[List Mode]

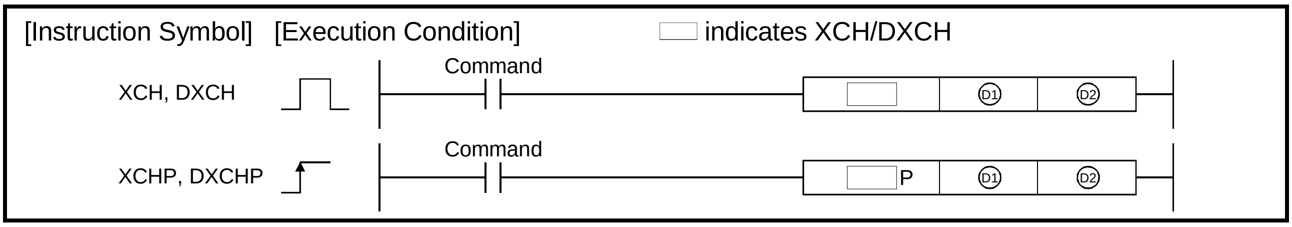
Steps	Instruction	Device
0	LD	X0A
1	FMOVP	K1X20 D100 K4
5	END	


X2F- -X2C X2B- -X28 X27- -X24 X23- -X20
Before execution
1 0 1 1 0 1 1 1 0 0 1 0 1 1 1 0
Ignored
After execution (transfer destination)
b15
b4
b3
b0
0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 D100
0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 D101
0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 D102
0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 D103
4 points
Becomes 0

QCPU				QnA	Q4AR
PLC CPU			Process CPU		
Basic	High Performance				
○	○		○	○	○

6.4.7 16-bit and 32-bit data exchanges (XCH, XCHP, DXCH, DXCHP)

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant K, H	Other
	Bit	Word		Bit	Word				
D1	○							—	
D2	○							—	



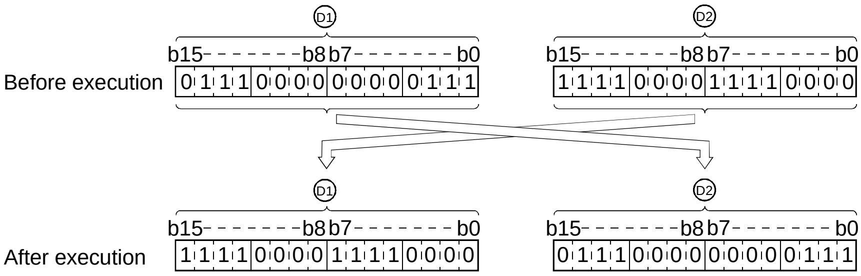
[Set Data]

Set Data	Meaning	Data Type
D1 D2	Head number of device storing data to be exchanged	BIN 16/32 bits

[Functions]

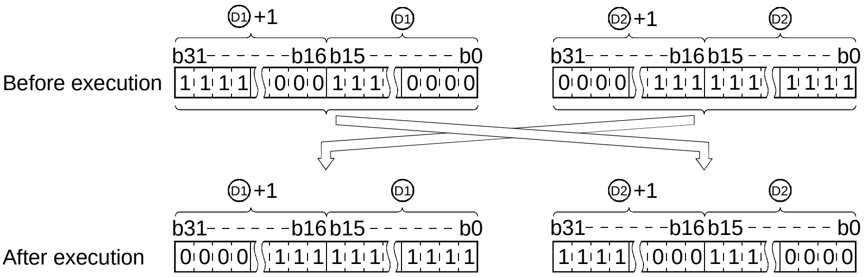
XCH

(1) Conducts 16-bit data exchange between D1 and D2.



DXCH

(1) Conducts 32-bit data exchange between D1+1, D1 and D2+1, D2.



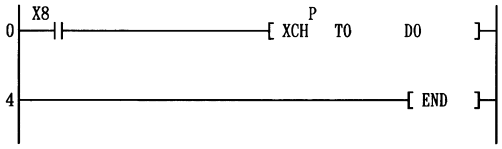
[Operation Errors]

(1) There are no errors associated with the XCH (P) and DXCH (P) instructions.

[Program Example]

(1) The following program exchanges the present value of T0 with the contents of D0 when X8 goes ON.

[Ladder Mode]

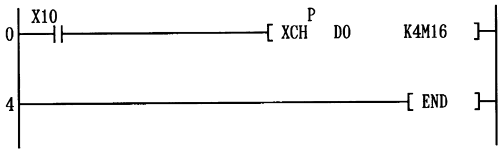


[List Mode]

Steps	Instruction	Device
0	LD	X8
1	XCHP	T0 D0
4	END	

(2) The following program exchanges the contents of D0 with the data from M16 to M31 when X10 goes ON.

[Ladder Mode]

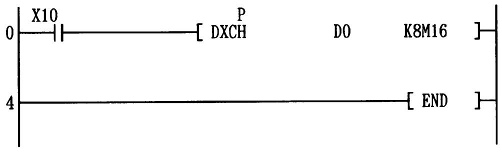


[List Mode]

Steps	Instruction	Device
0	LD	X10
1	XCHP	D0 K4M16
4	END	

(3) The following program exchanges the contents of D0 and D1 with the data at M16 to M47 when X10 goes ON.

[Ladder Mode]

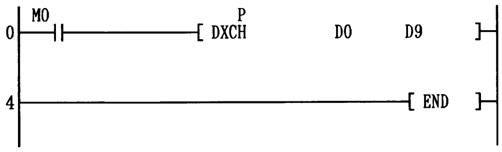


[List Mode]

Steps	Instruction	Device
0	LD	X10
1	DXCHP	D0 K8M16
4	END	

(4) The following program exchanges the contents of D0 and D1 with those of D9 and D10 when M0 goes ON.

[Ladder Mode]



[List Mode]

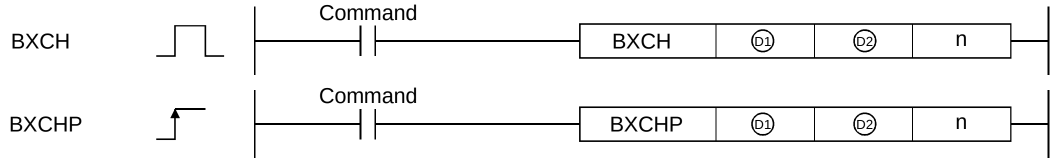
Steps	Instruction	Device
0	LD	M0
1	DXCHP	D0 D9
4	END	

QCPU			QnA	Q4AR
PLC CPU		Process CPU		
Basic	High Performance			
○	○	○	○	○

6.4.8 Block 16-bit data exchanges (BXCH, BXCHP)

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant K, H	Other
	Bit	Word		Bit	Word				
D1	—	○		—					—
D2	—	○		—					—
n	○	○		○					—

[Instruction Symbol] [Execution Condition]

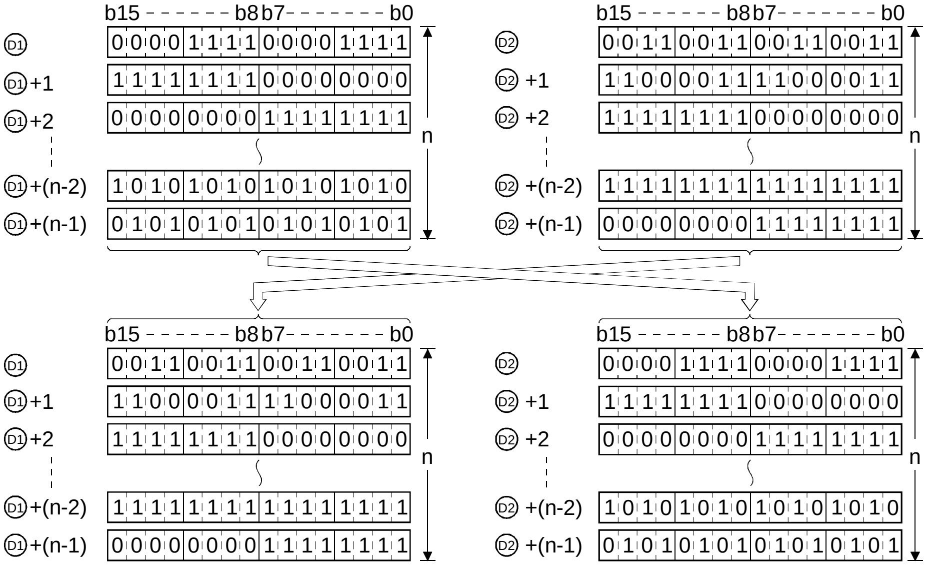


[Set Data]

Set Data	Meaning	Data Type
D1	Head number of device storing data to be exchanged	BIN 16 bits
D2		
n	Number of exchanges	

[Functions]

(1) Exchanges 16-bit data n-points from device designated by D1 and 16-bit data n-points from device designated by D2.


b15 - - - - - - b8 b7 - - - - - - - b0
D1 0000111100001111
D1+1 1111111100000000
D1+2 0000000011111111
D1+(n-2) 1010101010101010
D1+(n-1) 0101010101010101
n
b15 - - - - - - b8 b7 - - - - - - - b0
D2 0011001100110011
D2 +1 1100001111000011
D2 +2 1111111100000000
D2 +(n-2) 1111111111111111
D2 +(n-1) 0000000011111111
n
b15 - - - - - - b8 b7 - - - - - - - b0
D1 0011001100110011
D1+1 1100001111000011
D1+2 1111111100000000
D1+(n-2) 1111111111111111
D1+(n-1) 0000000011111111
n
b15 - - - - - - b8 b7 - - - - - - - b0
D2 0000111100001111
D2 +1 1111111100000000
D2 +2 0000000011111111
D2 +(n-2) 1010101010101010
D2 +(n-1) 0101010101010101
n

[Operation Errors]

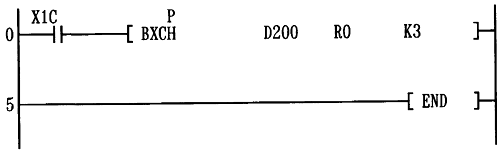
(1) In the following cases an operation error occurs, the error flag (SM0) turns ON, and an error code is stored at SD0.

- The range n-points from the Ⓓ1 or Ⓓ2 devices exceeds relevant device. (Error code: 4101)
- Ⓓ1 and Ⓓ2 devices are overlapping. (Error code: 4101)

[Program Example]

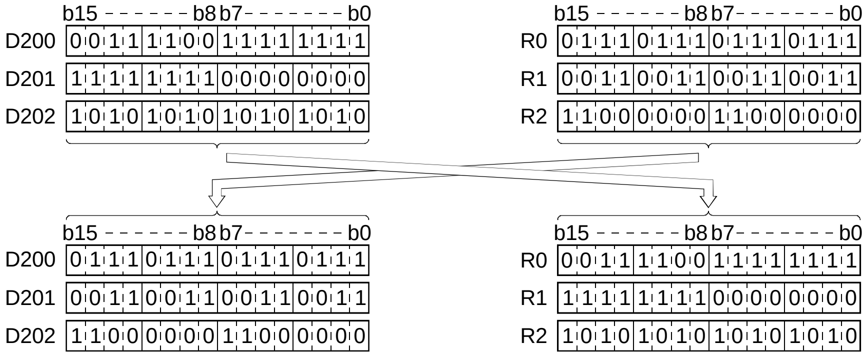
(1) The following program exchanges 16-bit data for 3 points from D200 for 16-bit data for 3 points from R0 when X1C goes ON.

[Ladder Mode]



[List Mode]

Steps	Instruction	Device
0	LD	X1C
1	BXCHP	D200 R0 K3
5	END	



	b15 - - - - - - b8	b7 - - - - - - - b0
D200	0011 1100	1111 1111
D201	1111 1111	0000 0000
D202	1010 1010	1010 1010

	b15 - - - - - - b8	b7 - - - - - - - b0
R0	0111 0111	0111 0111
R1	0011 0011	0011 0011
R2	1100 0000	1100 0000

	b15 - - - - - - b8	b7 - - - - - - - b0
D200	0111 0111	0111 0111
D201	0011 0011	0011 0011
D202	1100 0000	1100 0000

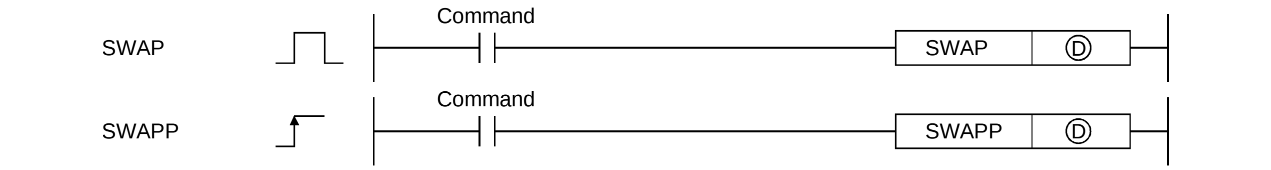
	b15 - - - - - - b8	b7 - - - - - - - b0
R0	0011 1100	1111 1111
R1	1111 1111	0000 0000
R2	1010 1010	1010 1010

QCPU			QnA	Q4AR
PLC CPU		Process CPU		
Basic	High Performance			
○	○	○	○	○

6.4.9 Upper and lower byte exchanges (SWAP, SWAPP)

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant K, H	Other
	Bit	Word		Bit	Word				
Ⓓ	○							—	

[Instruction Symbol] [Execution Condition]

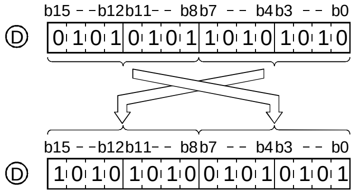


[Set Data]

Set Data	Meaning	Data Type
Ⓓ	Head number of device where data is stored	BIN 16 bits

[Functions]

(1) Exchanges the higher and lower 8 bits of the device designated by Ⓓ.


b15 - - b12 b11 - - b8 b7 - - b4 b3 - - b0
Ⓓ
0 1 0 1 0 1 0 1 1 0 1 0 1 0 1 0
b15 - - b12 b11 - - b8 b7 - - b4 b3 - - b0
Ⓓ
1 0 1 0 1 0 1 0 0 1 0 1 0 1 0 1

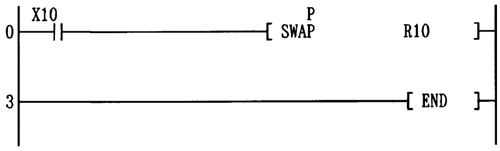
[Operation Errors]

(1) There are no operation errors associated with the SWAP(P) instruction.

[Program Example]

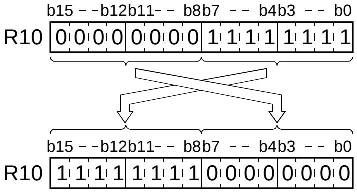
(1) The following program exchanges the higher 8 bits and lower 8 bits of R10 when X10 goes ON.

[Ladder Mode]



[List Mode]

Steps	Instruction	Device
0	LD	X10
1	SWAPP	R10
3	END	


b15 - - b12 b11 - - b8 b7 - - b4 b3 - - b0
R10
0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1
b15 - - b12 b11 - - b8 b7 - - b4 b3 - - b0
R10
1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0

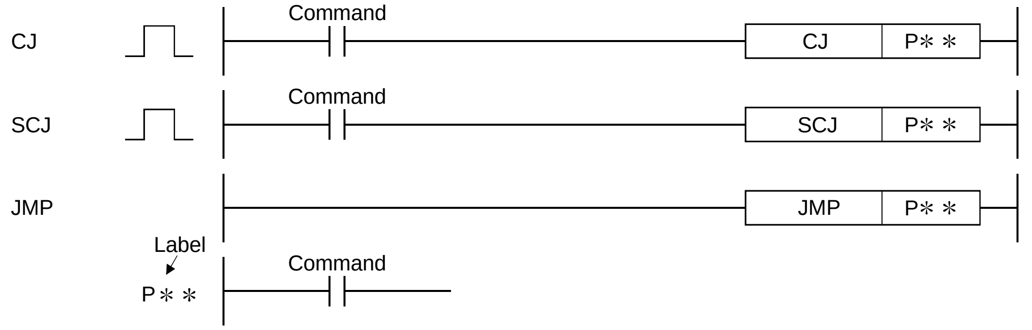
QCPU			QnA	Q4AR
PLC CPU		Process CPU		
Basic	High Performance			
○	○	○	○	○

6.5 Program Branch Instruction

6.5.1 Pointer branch instructions (CJ, SCJ, JMP)

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant K, H	Other
	Bit	Word		Bit	Word				P
P	—								○

[Instruction Symbol] [Execution Condition]



[Set Data]

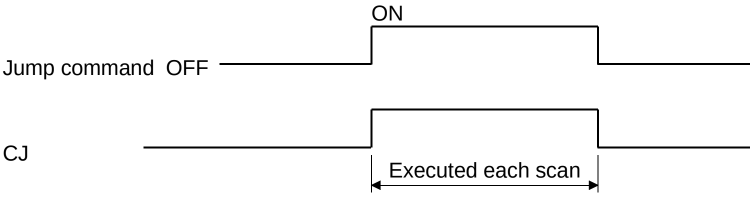
Set Data	Meaning	Data Type
P＊＊	Pointer number of jump destination	Device name

[Functions]

CJ

(1) Executes program of designated pointer number within the same program file when jump command is ON.

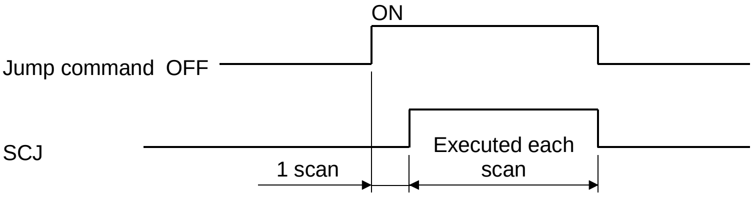
(2) Executes next step in program when jump command is OFF.



SCJ

(1) Executes program of designated pointer number within the same program file from next scan when jump command goes from OFF to ON.

(2) Executes next step in program when jump command is OFF or when it goes from ON to OFF.



JMP

(1) Unconditionally executes program of designated pointer number within the same program file.

POINTS

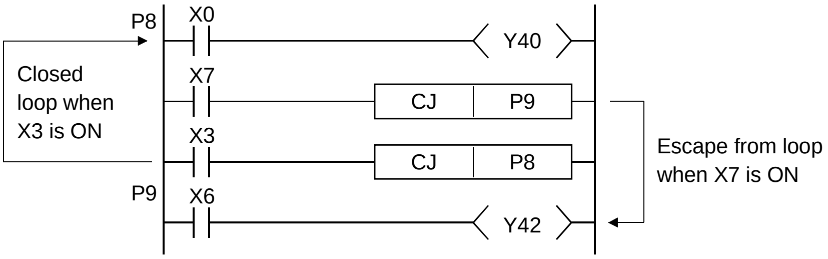
Note the following points when using the jump instruction.

(1) After the timer coil has gone ON, accurate measurements cannot be made if there is an attempt to jump the timer of a coil that has been turned ON using the CJ, SCJ or JMP instructions.

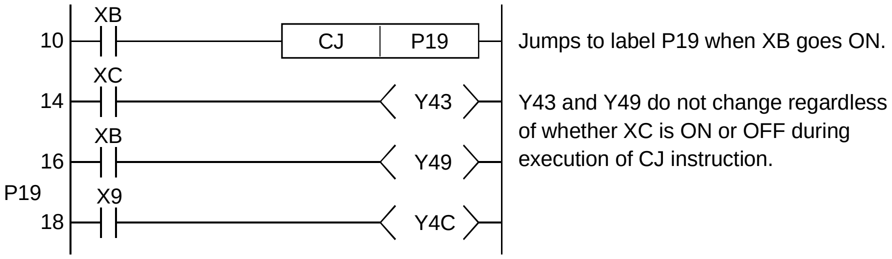
(2) Scan time is shortened if the CJ, SCJ or JMP instruction is used to force a jump to the OUT instruction.

(3) Scan time is shortened if the CJ, SCJ or JMP instruction is used to force a jump to the rear.

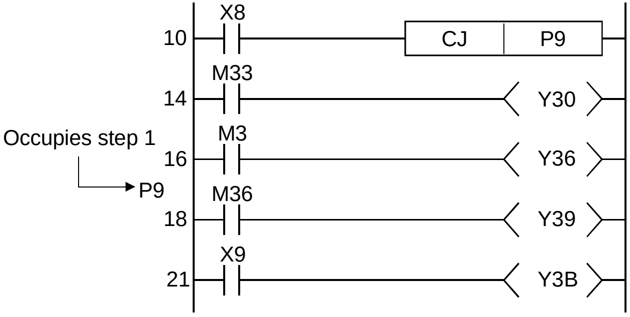
(4) The CJ, SCJ, and JMP instructions can be used to jump to a step prior to the step currently being executed.
However, it is necessary to consider methods to get out of the loop so that the watchdog timer does not time out in the process.



(5) The device to which a jump has been made with CJ, SCJ or JMP does not change.


10
XB
CJ P19
Jumps to label P19 when XB goes ON.
14
XC
Y43
Y43 and Y49 do not change regardless of whether XC is ON or OFF during execution of CJ instruction.
16
XB
Y49
P19
18
X9
Y4C

(6) The label (P✻) occupies step 1.



(7) Jump instructions can be used only for pointer numbers within the same program file.

(8) If a jump is made to a pointer number inside the skip range during a skip operation, program execution will be taken up following the pointer number of the jump destination.

[Operation Errors]

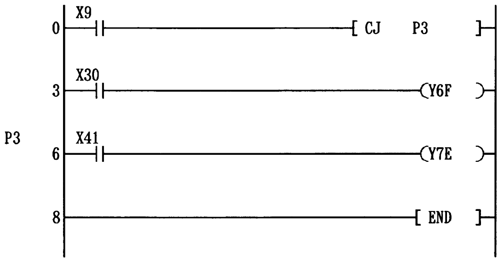
(1) In the following cases an operation is returned, the error flag (SM0) goes ON, and the error code is stored at SD0.

- The pointer number designated does not come prior to the END instruction. (Error code: 4210)
- A pointer number which is not in use as a label in the same program has been designated. (Error code: 4210)
- A common pointer has been designated. (Error code: 4210)

[Program Example]

(1) The following program jumps to P3 when X9 goes ON.

[Ladder Mode]

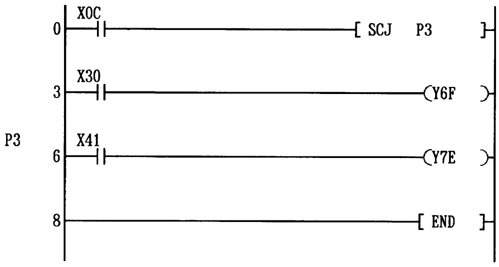


[List Mode]

Steps	Instruction	Device
0	LD	X9
1	CJ	P3
3	LD	X30
4	OUT	Y6F
5		P3
6	LD	X41
7	OUT	Y7E
8	END	

(2) The following program jumps to P3 from the next scan after XC goes ON.

[Ladder Mode]



[List Mode]

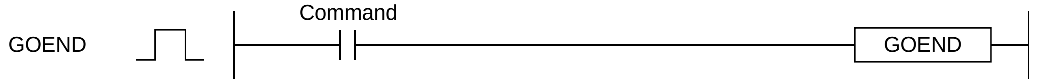
Steps	Instruction	Device
0	LD	X0C
1	SCJ	P3
3	LD	X30
4	OUT	Y6F
5		P3
6	LD	X41
7	OUT	Y7E
8	END	

QCPU			QnA	Q4AR
PLC CPU		Process CPU		
Basic	High Performance			
○	○	○	○	○

6.5.2 Jump to END (GOEND)

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant K, H	Other
	Bit	Word		Bit	Word				
—	—								

[Instruction Symbol] [Execution Condition]



[Functions]

(1) Jumps to FEND or END instruction in the same program file.

[Operation Errors]

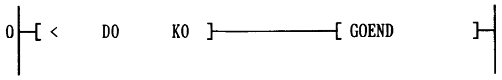
(1) In the following cases an operation error occurs, the error flag (SM0) turns ON, and an error code is stored at SD0.

- A GOEND instruction has been executed after the execution of a CALL, ECALL instruction, and prior to the execution of the RET instruction. (Error code: 4211)
- A GOEND instruction has been executed after the execution of a FOR instruction, and prior to the execution of the NEXT instruction. (Error code: 4200)
- A GOEND instruction has been executed during an interrupt program but prior to the execution of the IRET instruction. (Error code: 4221)
- A GOEND instruction was executed between the CHKCIR and CHKEND instruction block. (Error code: 4230)
- A GOEND instruction was executed between the IX and IXEND instruction block. (Error code: 4231)

[Program Example]

(1) The following program jumps to the END instruction if D0 holds a negative number.

[Ladder Mode]



[List Mode]

Steps	Instruction	Device
0	LD<	D0 K0
3	GOEND	

QCPU			QnA	Q4AR
PLC CPU		Process CPU		
Basic	High Performance			
○	○	○	○	○

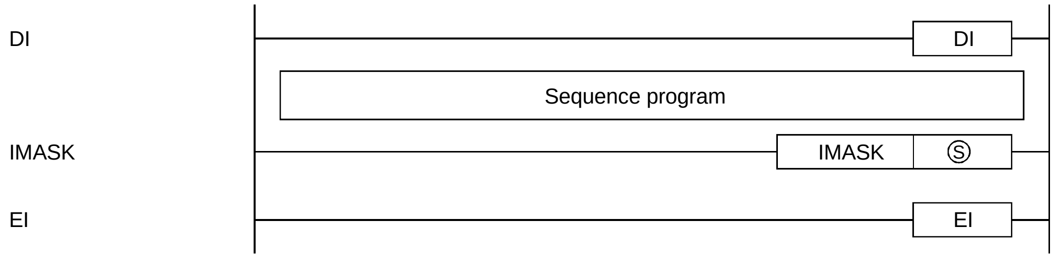
6.6 Program Execution Control Instructions

6.6.1 Interrupt disable/enable instructions, interrupt program mask (DI, EI, IMASK)

(1) When Basic model QCPU is used

Set Data	Usable Devices									
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant K, H	Other
	Bit	Word		Bit	Word				
Ⓢ	–	○		–					

[Instruction Symbol] [Execution Condition]



[Set Data]

Set Data	Meaning	Data Type
Ⓢ	Interrupt mask data or head number of device where interrupt mask data is being stored	BIN 16 bits

[Functions]

DI

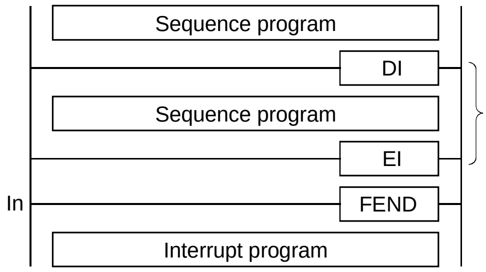
(1) Disables the execution of an interrupt program until the EI instruction has been executed, even if a start cause for the interrupt program occurs.

(2) A DI state is entered when power is turned ON or when the system has been reset.

EI

(1) The EI instruction is used to clear the interrupt disable state resulting from the execution of the DI instruction, and to create a state in which the interrupt program designated by the interrupt pointer number certified by the IMASK instruction can be executed.
When the IMASK instruction is not executed, I32 to I47 are disabled.

(2) Be sure to execute the EI instruction before executing a periodic program.



Even though an interrupt condition might be generated between the DI and EI instructions, the interrupt program will be held until the entire cycle from DI to EI has been processed.

IMASK

(1) Enables/disables the execution of the interrupt program marked by the designated interrupt pointer by using the bit pattern of 8 points from the device designated by Ⓢ .

- 1 (ON) ......Interrupt program execution enabled
- 0 (OFF).....Interrupt program execution disabled

(2) The interrupt pointer numbers corresponding to the individual bits are as shown below:

	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Ⓢ	I15	I14	I13	I12	I11	I10	I9	I8	I7	I6	I5	I4	I3	I2	I1	I0
Ⓢ + 1	I31	I30	I29	I28	I27	I26	I25	I24	I23	I22	I21	I20	I19	I18	I17	I16
Ⓢ + 2	I47	I46	I45	I44	I43	I42	I41	I40	I39	I38	I37	I36	I35	I34	I33	I32
Ⓢ + 3	I63	I62	I61	I60	I59	I58	I57	I56	I55	I54	I53	I52	I51	I50	I49	I48
Ⓢ + 4	I79	I78	I77	I76	I75	I74	I73	I72	I71	I70	I69	I68	I67	I66	I65	I64
Ⓢ + 5	I95	I94	I93	I92	I91	I90	I89	I88	I87	I86	I85	I84	I83	I82	I81	I80
Ⓢ + 6	I111	I110	I109	I108	I107	I106	I105	I104	I103	I102	I101	I100	I99	I98	I97	I96
Ⓢ + 7	I127	I126	I125	I124	I123	I122	I121	I120	I119	I118	I117	I116	I115	I114	I113	I112

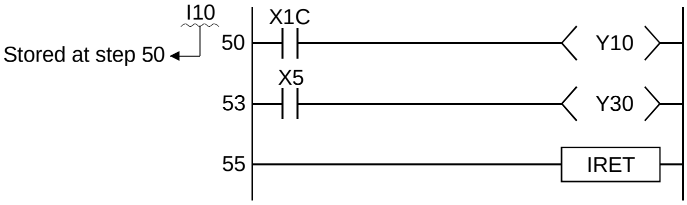
(3) When the power is turned ON or when the CPU module has been reset with the execution of interrupt programs I0 to I31 is enabled.

(4) The statuses of devices Ⓢ, Ⓢ +1, Ⓢ +2, and Ⓢ +3 to Ⓢ +7 are stored in SD715 to SD717 and SD781 to SD785 (storage area for IMASK instruction mask pattern).

(5) Although the special registers are separated as SD715 to SD717 and SD781 to SD785, device numbers should be designated as Ⓢ to Ⓢ +7 successively.

POINTS

(1) An interrupt pointer occupies 1 step.



(2) Refer to the Basic model QCPU (Q mode) User's Manual (Function Explanation, Program Fundamentals) for information on interrupt conditions.

(3) The DI state (interrupt disabled) is active during the execution of an interrupt program. Do not insert EI instructions in interrupt programs to attempt the execution of multiple interrupts, with interrupt programs running inside interrupt programs.

(4) If there are EI and DI instructions within a master control, these instructions will be executed regardless of the execution/non-execution status of the MC instruction.

[Operation Errors]

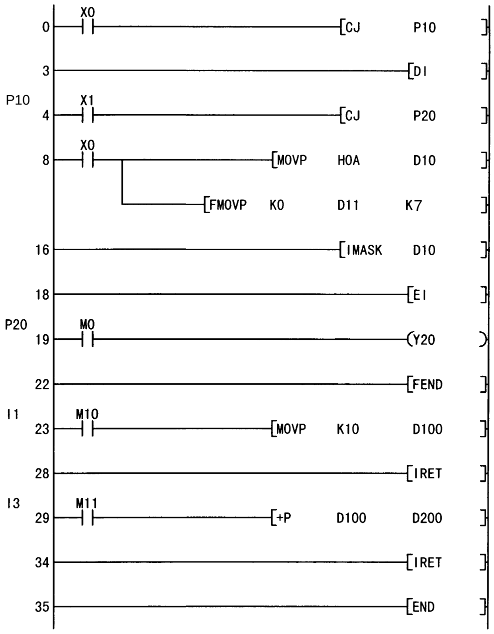
(1) There are no operation errors associated with the DI and EI instructions.

(2) There are no operation errors associated with the IMASK instruction.

[Program Example]

(1) The following program is designed to enable the execution of only the interrupt programs having the interrupt pointer numbers I1 and I3 while X0 is ON.

[Ladder Mode]



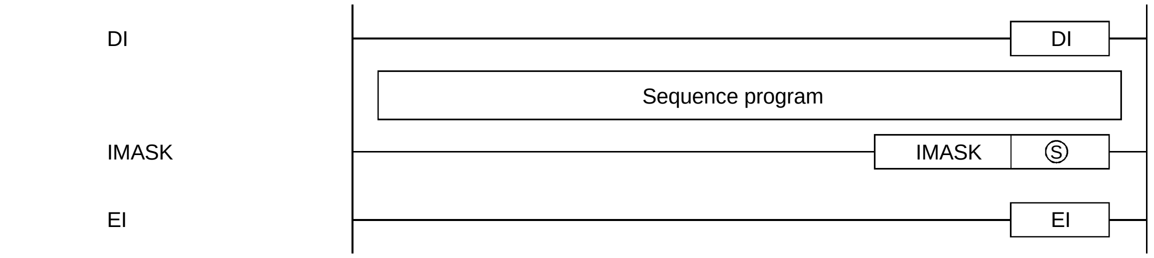
[List Mode]

Steps	Instruction	Device
0	LD	X0
1	CJ	P10
3	DI	
4	P10	
5	LD	X1
6	CJ	P20
8	LD	X0
9	MOVP	H0A D10
12	FMOVP	K0 D11 K7
16	IMASK	D10
18	EI	
19	P20	
20	LD	M0
21	OUT	Y20
22	FEND	
23	I1	
24	LD	M10
25	MOVP	K10 D100
28	IRET	
29	I3	
30	LD	M11
31	+P	D100 D200
34	IRET	
35	END	

(2) When the High Performance model QCPU/Process CPU is used

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant K, H	Other
	Bit	Word		Bit	Word				
Ⓢ	—	○		—					

[Instruction Symbol] [Execution Condition]



[Set Data]

Set Data	Meaning	Data Type
Ⓢ	Interrupt mask data or head number of device where interrupt mask data is being stored	BIN 16 bits

[Functions]

DI

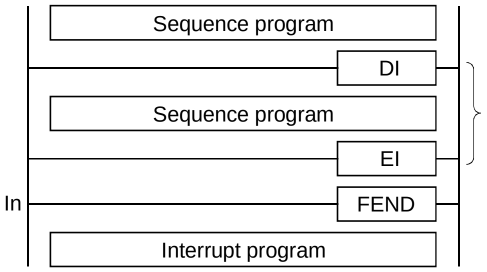
(1) Disables the execution of an interrupt program until the EI instruction has been executed, even if a start cause for the interrupt program occurs.

(2) A DI state is entered when power is turned ON or when the system has been reset.

EI

(1) The EI instruction is used to clear the interrupt disable state resulting from the execution of the DI instruction, and to create a state in which the interrupt program designated by the interrupt pointer number certified by the IMASK instruction can be executed.
When the IMASK instruction is not executed, I32 to I47 are disabled.

(2) Be sure to execute the EI instruction before executing a periodic program.



Even though an interrupt condition might be generated between the DI and EI instructions, the interrupt program will be held until the entire cycle from DI to EI has been processed.

IMASK

(1) Enables/disables the execution of the interrupt program marked by the designated interrupt pointer by using the bit pattern of 16 points from the device designated by Ⓢ .
- 1 (ON) ......Interrupt program execution enabled
- 0 (OFF).....Interrupt program execution disabled

(2) The interrupt pointer numbers corresponding to the individual bits are as shown below:

	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Ⓢ	I15	I14	I13	I12	I11	I10	I9	I8	I7	I6	I5	I4	I3	I2	I1	I0
Ⓢ+ 1	I31	I30	I29	I28	I27	I26	I25	I24	I23	I22	I21	I20	I19	I18	I17	I16
Ⓢ+ 2	I47	I46	I45	I44	I43	I42	I41	I40	I39	I38	I37	I36	I35	I34	I33	I32
Ⓢ+ 3	I63	I62	I61	I60	I59	I58	I57	I56	I55	I54	I53	I52	I51	I50	I49	I48
Ⓢ+ 4	I79	I78	I77	I76	I75	I74	I73	I72	I71	I70	I69	I68	I67	I66	I65	I64
Ⓢ+ 5	I95	I94	I93	I92	I91	I90	I89	I88	I87	I86	I85	I84	I83	I82	I81	I80
Ⓢ+ 6	I111	I110	I109	I108	I107	I106	I105	I104	I103	I102	I101	I100	I99	I98	I97	I96
Ⓢ+ 7	I127	I126	I125	I124	I123	I122	I121	I120	I119	I118	I117	I116	I115	I114	I113	I112
Ⓢ+ 8	I143	I142	I141	I140	I139	I138	I137	I136	I135	I134	I133	I132	I131	I130	I129	I128
Ⓢ+ 9	I159	I158	I157	I156	I155	I154	I153	I152	I151	I150	I149	I148	I147	I146	I145	I144
Ⓢ+10	I175	I174	I173	I172	I171	I170	I169	I168	I167	I166	I165	I164	I163	I162	I161	I160
Ⓢ+11	I191	I190	I189	I188	I187	I186	I185	I184	I183	I182	I181	I180	I179	I178	I177	I176
Ⓢ+12	I207	I206	I205	I204	I203	I202	I201	I200	I199	I198	I197	I196	I195	I194	I193	I192
Ⓢ+13	I223	I222	I221	I220	I219	I218	I217	I216	I215	I214	I213	I212	I211	I210	I209	I208
Ⓢ+14	I239	I238	I237	I236	I235	I234	I233	I232	I231	I230	I229	I228	I227	I226	I225	I224
Ⓢ+15	I255	I254	I253	I252	I251	I250	I249	I248	I247	I246	I245	I244	I243	I242	I241	I240

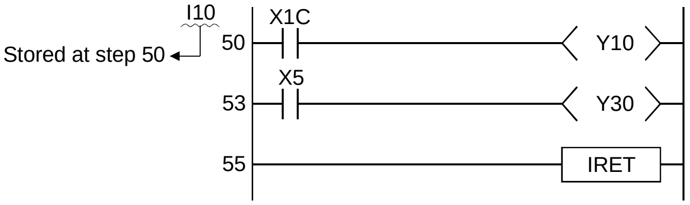
(3) When the power is turned ON or when the CPU module has been reset, the execution of interrupt programs I0 to I31,I48 to I255 is enabled, and the execution of interrupt programs I32 to I47 is disabled.

(4) The statuses of devices Ⓢ, Ⓢ +1, Ⓢ +2, and Ⓢ +3 to Ⓢ +15 are stored in SD715 to SD717 and SD781 to SD793 (storage area for IMASK instruction mask pattern).

(5) Although the special registers are separated as SD715 to SD717 and SD781 to SD793, device numbers should be designated as Ⓢ to Ⓢ +15 successively.

POINTS

(1) An interrupt pointer occupies 1 step.



(2) Refer to the User’s Manual (Function Explanation, Program Fundamentals) of the CPU module in use for interrupt conditions.

(3) The DI state (interrupt disabled) is active during the execution of an interrupt program. Do not insert EI instructions in interrupt programs to attempt the execution of multiple interrupts, with interrupt programs running inside interrupt programs.

(4) If there are EI and DI instructions within a master control, these instructions will be executed regardless of the execution/non-execution status of the MC instruction.

[Operation Errors]

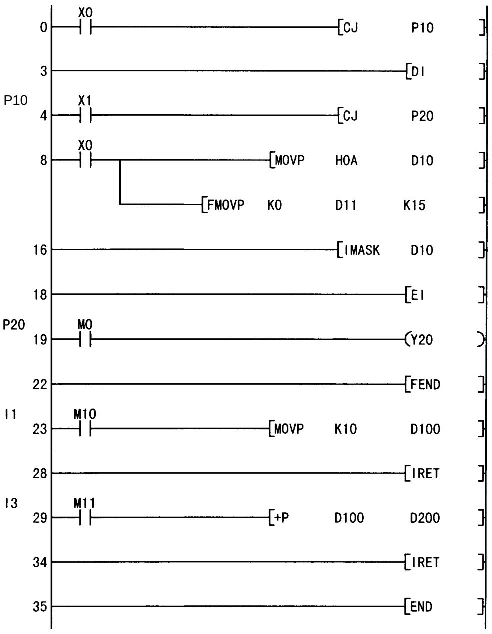
(1) There are no operation errors associated with the DI and EI instructions.

(2) There are no operation errors associated with the IMASK instruction.

[Program Example]

(1) The following program creates an execution enabled state for the interrupt program marked by the interrupt pointer number when X0 is ON.

[Ladder Mode]



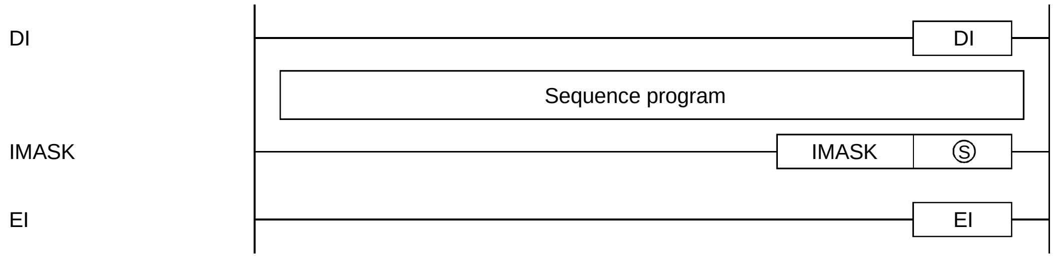
[List Mode]

Steps	Instruction	Device
0	LD	X0
1	CJ	P10
3	DI	
4	P10	
5	LD	X1
6	CJ	P20
8	LD	X0
9	MOVP	H0A D10
12	FMOVP	K0 D11 K15
16	IMASK	D10
18	EI	
19	P20	
20	LD	M0
21	OUT	Y20
22	FEND	
23	I1	
24	LD	M10
25	MOVP	K10 D100
28	IRET	
29	I3	
30	LD	M11
31	+P	D100 D200
34	IRET	
35	END	

(3) When QnACPU is used

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant K, H	Other
	Bit	Word		Bit	Word				
Ⓢ	—	○		—					

[Instruction Symbol] [Execution Condition]



[Set Data]

Set Data	Meaning	Data Type
Ⓢ	Interrupt mask data or head number of device where interrupt mask data is being stored	BIN 16 bits

[Functions]

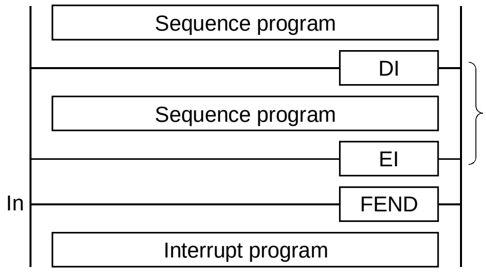
DI

(1) Disables the execution of an interrupt program until the EI instruction has been executed, even if a start cause for the interrupt program occurs.

(2) A DI state is entered when power is turned ON or when the system has been reset.

EI

The EI instruction is used to clear the interrupt disable state resulting from the execution of the DI instruction, and to create a state in which the interrupt program designated by the interrupt pointer number certified by the IMASK instruction can be executed.



Even though an interrupt condition might be generated between the DI and EI instructions, the interrupt program will be held until the entire cycle from DI to EI has been processed.

IMASK

(1) Enables or disables the execution of the interrupt program marked by the designated interrupt pointer by use of the bit pattern in the three points from the device designated by Ⓢ .

- 1 (ON) .... Interrupt program execution enabled
- 0 (OFF)........ Interrupt program execution disabled

(2) The interrupt pointer numbers corresponding to the individual bits are as shown below:

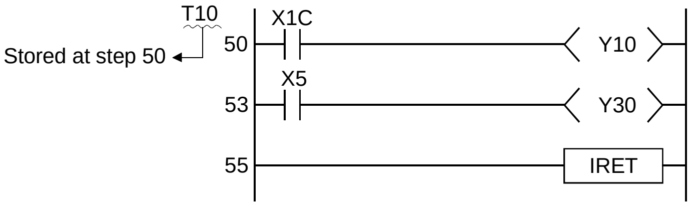
	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Ⓢ	I15	I14	I13	I12	I11	I10	I9	I8	I7	I6	I5	I4	I3	I2	I1	I0
Ⓢ+ 1	I31	I30	I29	I28	I27	I26	I25	I24	I23	I22	I21	I20	I19	I18	I17	I16
Ⓢ+ 2	I47	I46	I45	I44	I43	I42	I41	I40	I39	I38	I37	I36	I35	I34	I33	I32

(3) When the power is turned ON, or when the CPU module has been reset, interrupt programs from I0 to I31 are in the execution enabled state, and interrupt programs from I32 to I47 are in the execution disabled state.

(4) The statuses of the Ⓢ, Ⓢ +1, and Ⓢ +2 devices are stored from SD715 to SD717 (the IMASK instruction mask pattern storage area).

POINTS

(1) An interrupt pointer occupies 1 step.



(2) Refer to the QnACPU Programming Manual (Fundamentals) for interrupt conditions.

(3) The DI state (interrupt disabled) is active during the execution of an interrupt program. Do not insert EI instructions in interrupt programs to attempt the execution of multiple interrupts, with interrupt programs running inside interrupt programs.

(4) If there are EI and DI instructions within a master control, these instructions will be executed regardless of the execution/non-execution status of the MC instruction.

[Operation Errors]

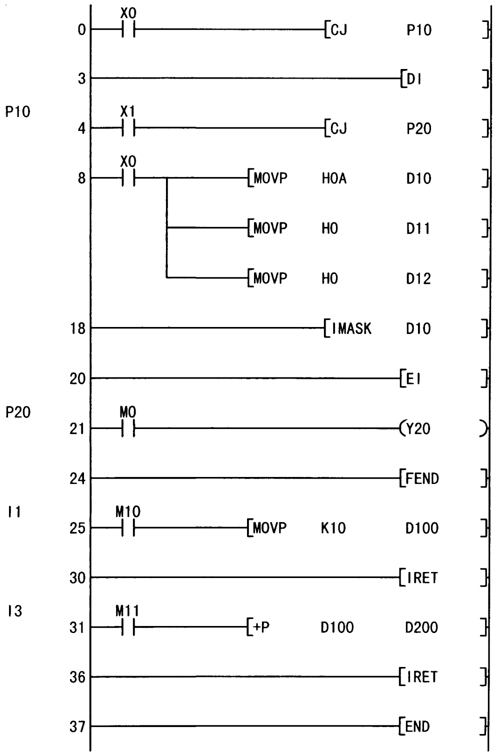
(1) There are no operation errors associated with the DI and EI instructions.

(2) There are no operation errors associated with the IMASK instruction.

[Program Example]

(1) The following program creates an execution enabled state for the interrupt program marked by the interrupt pointer number when X0 is ON.

[Ladder Mode]



[List Mode]

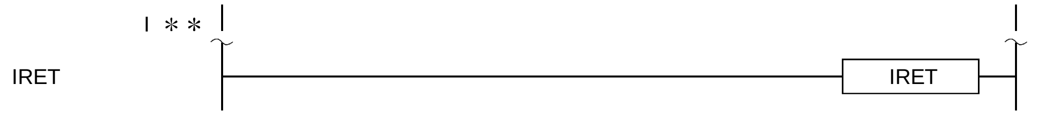
Steps	Instruction	Device	
0	LD	X0	
1	CJ	P10	
3	DI		
4	P10		
5	LD	X1	
6	CJ	P20	
8	LD	X0	
9	MOVP	H0A	D10
12	MOVP	H0	D11
15	MOVP	H0	D12
18	IMASK	D10	
20	EI		
21	P20		
22	LD	M0	
23	OUT	Y20	
24	FEND		
25	I1		
26	LD	M10	
27	MOVP	K10	D100
30	IRET		
31	I3		
32	LD	M11	
33	+P	D100	D200
36	IRET		
37	END		

QCPU			QnA	Q4AR
PLC CPU		Process CPU		
Basic	High Performance			
○	○	○	○	○

6.6.2 Recovery from interrupt programs (IRET)

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant K, H	Other
	Bit	Word		Bit	Word				
—	—								

[Instruction Symbol] [Execution Condition]



[Functions]

(1) Indicates the completion of interrupt program processing.

(2) Returns to sequence program processing following the execution of the IRET instruction.

[Operation Errors]

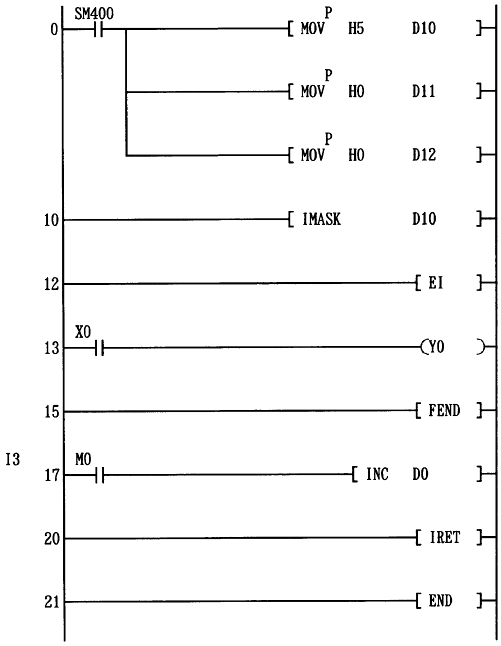
(1) In the following cases an operation error occurs, the error flag (SM0) turns ON, and an error code is stored at SD0.

- There is no pointer corresponding to the interrupt number. (Error code: 4220)
- The IRET instruction has been issued prior to the execution of the interrupt program. (Error code: 4223)
- An END, FEND, GOEND, or STOP instruction as been executed after the generation of an interrupt and prior to the execution of the IRET instruction. (Error code: 4221)

[Program Example]

(1) The following program adds 1 to D0 if M0 is ON when the number 3 interrupt is generated.

[Ladder Mode]



[List Mode]

Steps	Instruction	Device
0	LD	SM400
1	MOVP	H5 D10
4	MOVP	H0 D11
7	MOVP	H0 D12
10	IMASK	D10
12	EI	
13	LD	X0
14	OUT	Y0
15	FEND	
16		I3
17	LD	M0
18	INC	D0
20	IRET	
21	END	

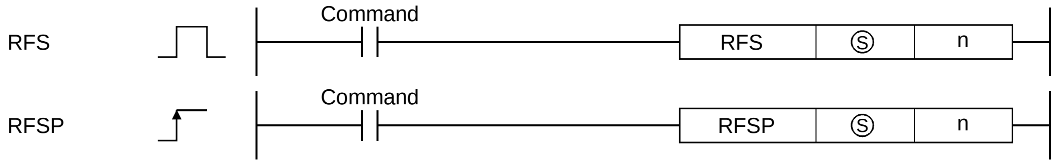
QCPU			QnA	Q4AR
PLC CPU		Process CPU		
Basic	High Performance			
○	○	○	○	○

6.7 I/O Refresh Instructions

6.7.1 I/O Refresh (RFS, RFSP)

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant K, H	Other
	Bit	Word		Bit	Word				
Ⓢ	○ (Only X, Y)	—							—
n	○	○							—

[Instruction Symbol] [Execution Condition]



[Set Data]

Set Data	Meaning	Data Type
Ⓢ	Head device number of the device that will conduct refresh operation	Bit
n	Number of points to be refreshed	BIN 16 bits

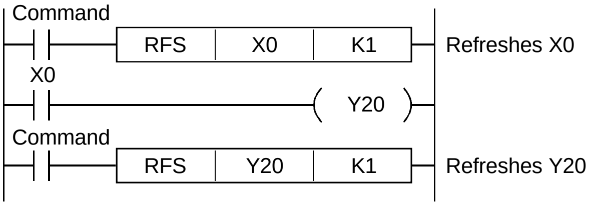
[Functions]

(1) Refreshes only the device being scanned during a scan, and functions to fetch input from external sources or to output data to an output module.

(2) Fetching of input from or sending output to an external source is conducted in batch only after the execution of an END instruction, so it is not possible to output a pulse signal to an outside source during the execution of a scan.
When a refresh operation is conducted, inputs (X) or outputs (Y) of the device numbers relevant to the program being executed are forcibly refreshed, so it is possible to output a pulse signal to an external source during a scan.

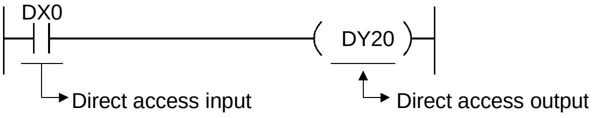
(3) Use direct access inputs (DX) or direct access outputs (DY) to refresh inputs (X) or outputs (Y) in 1-point units.

[Program based on the RFS instruction]



⇩

[Program based on DX and DY]



[Operation Errors]

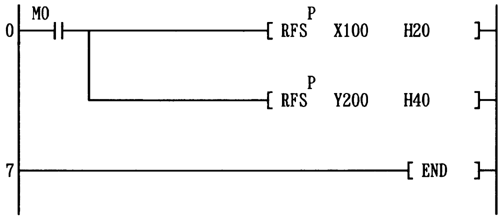
(1) In the following cases an operation error occurs, the error flag (SM0) turns ON, and an error code is stored at SD0.

- The range n points from the device designated by Ⓢ exceeds the proximate I/O range.

[Program Example]

(1) The following program refreshes X100 to X11F and Y200 to Y23F when M0 goes ON.

[Ladder Mode]



[List Mode]

Steps	Instruction	Device
0	LD	M0
1	RFSP	X100 H20
4	RFSP	Y200 H40
7	END	

QCPU			QnA	Q4AR
PLC CPU		Process CPU		
Basic	High Performance			
×	○	○	○	○

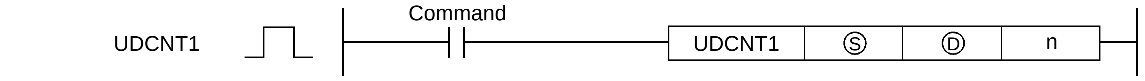
6.8 Other Convenient Instructions

6.8.1 Count 1-phase input up or down (UDCNT1)

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant K, H	Other
	Bit	Word		Bit	Word				
Ⓢ	○ (Only X)	–	–	–					–
Ⓓ	–	△ ∗ (Only C)	–	–					–
n	△ ∗	△ ∗	△ ∗	○					–

∗: Local devices and the file registers set for individual programs cannot be used.

[Instruction Symbol] [Execution Condition]



[Set Data]

Set Data	Meaning	Data Type
Ⓢ	• Ⓢ +0: Input number for count input • Ⓢ +1: For setting count upper down • OFF : Count up (add numbers when counting) • ON : Count down (subtract numbers when counting)	Bit
Ⓓ	• Number of counter that will perform count on UDCNT1 instruction	Word
n	• Set value	BIN16

[Functions]

(1) When the input designated at Ⓢ goes from OFF to ON, the present value of the counter designated at Ⓓ will be updated.

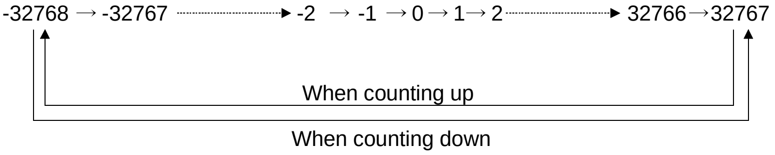
(2) The direction of the count is determined by the ON/OFF status of the input designated by Ⓢ +1.

- OFF : Count up (counts by adding to the present value)
- ON : Count down (counts by subtracting from the present value)

(3) Count processing is conducted as described below:

- When the count is going up, the counter contact designated at Ⓓ goes ON when the present value becomes identical with the setting value designated by n.
  However, the present value count will continue even when the contact of the counter designated at Ⓓ goes ON.(See Program Example (1))
- When the count is going down, the counter for the contact designated at Ⓓ goes OFF when the present value reaches the set value minus 1. (See Program Example (1))
- The counter designated at Ⓓ is a ring counter.
  If it is counting up when the present value is 32767, the present value will become -32768.
  Further, if it is counting down when the present value is -32768, the present value will become 32767.

The count processing performed on the present value is as shown below:



(4) Count processing based on the UDCNT1 instruction starts the count when the count command goes from OFF to ON, and suspends the count when it goes from ON to OFF.
When the count command goes from OFF to ON once again, the count is restarted from the value in effect when it was suspended.

(5) The RST instruction clears the present value of the counter designated at Ⓓ and turns the contact OFF.

POINTS

(1) The UDCNT1 instruction registers the argument device data to the work area of the CPU module and the actual counting operation is processed as a system interrupt.
(The device data registered to the work area of the CPU module are cleared when the command input is turned OFF or when the CPU module is STOPped and then RUN.)
Therefore, to count pulses, it is necessary to provide their ON and OFF time as long as the interrupt time of the CPU module or longer.
The interrupt time of individual CPU module is shown below:

CPU module Type Name	Interrupt Time
High Performance model QCPU, Process CPU	1ms
QnACPU	5ms

(2) The setting values cannot be changed during a count based on the UDCNT1 instruction (while the command input is ON)).
To change the setting values, first turn the command input off.

(3) Counters which have been designated by the UDCNT1 instruction cannot be used by other instructions. If they are used by other instructions, they will not be capable of returning an accurate count.

(4) The UDCNT1 instruction can be used as many as 6 times within all the programs being executed.
The seventh and the subsequent UDCNT1 instructions are not processed.

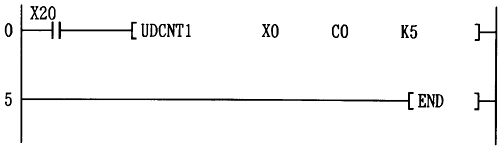
[Operation Errors]

(1) There are no operation errors associated with the UDCNT1 instruction.

[Program Example]

(1) This program uses C0 (up and down counter) to count the number of times X0 goes from off to ON after X20 has gone ON.

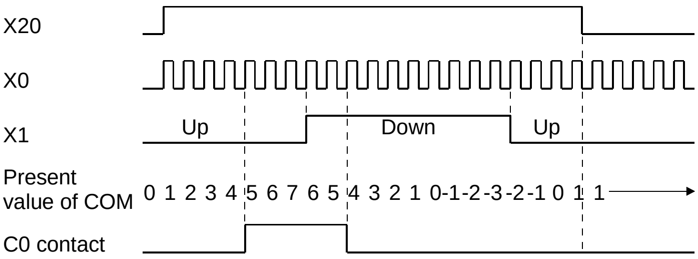
[Ladder Mode]



[List Mode]

Steps	Instruction	Device
0	LD	X20
1	UDCNT1	X0 C0 K5
5	END	

[Operation]



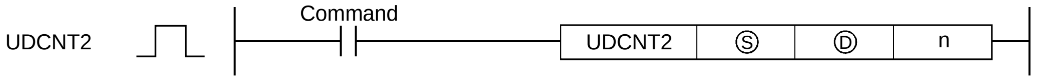
QCPU			QnA	Q4AR
PLC CPU		Process CPU		
Basic	High Performance			
×	○	○	○	○

6.8.2 Counter 2-phase input up or down (UDCNT2)

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant K, H	Other
	Bit	Word		Bit	Word				
Ⓢ	○ (Only X)	–	–	–					–
Ⓓ	–	△ ＊ (Only C)	–	–					–
n	△ ＊	△ ＊	△ ＊	○					–

＊: Local devices and the file registers set for individual programs cannot be used.

[Instruction Symbol] [Execution Condition]



[Set Data]

Set Data	Meaning	Data Type
Ⓢ	• Input number for count input: Ⓢ +0 (A phase pulse) • Input number for count input: Ⓢ +1 (B phase pulse)	Bit
Ⓓ	• Number of counter that will perform count onUDCNT2 instruction	Word
n	• Set value	BIN16

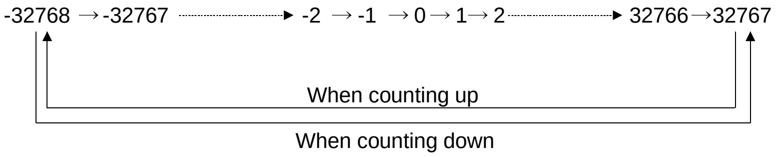
[Functions]

(1) The present value of the counter designated by Ⓓ is updated depending on the status of the input designated by Ⓢ (A phase pulse) and the status of the input designated by Ⓢ +1 (B phase pulse).

(2) Direction of the count is determined in the following manner:
- When Ⓢ is ON, if Ⓢ +1 goes from OFF to ON, count up operation is performed (values are added to the present value of the counter).
- When Ⓢ is ON, if Ⓢ +1 goes from ON to OFF, count down operation is performed (values are subtracted from the present value of the counter).
- No count operation is performed if Ⓢ is OFF.

(3) Count processing is conducted as described below:
- When the count is going up, the counter contact designated at Ⓓ goes ON when the present value becomes identical with the setting value designated by n.
  However, the present value count will continue even when the contact of the counter designated at Ⓓ goes ON. (See Program Example (1))
- When the count is going down, the counter for the contact designated at Ⓓ goes OFF when the present value reaches the setting value minus 1. (See Program Example (1))
- The counter designated at Ⓓ is a ring counter.
  If it is counting up when the present value is 32767, the present value will become -32768.
  Further, if it is counting down when the present value is -32768, the present value will become 32767.

The count processing performed on the present value is as shown below:



(4) Count processing conducted according to the UDCNT2 instruction begins when the count command goes from OFF to ON, and is suspended when it goes from ON to OFF.
When the count command goes from OFF to ON once again, the count is restarted from the value in effect when it was suspended.

(5) The RST instruction clears the present value of the counter designated at Ⓓ and turns the contact OFF.

POINTS

(1) The UDCNT2 instruction registers the argument device data to the work area of the CPU module and the actual counting operation is processed as a system interrupt.
(The device data registered to the work area of the CPU module are cleared when the command input is turned OFF or when the CPU module is STOPped and then RUN.)
Therefore, to count pulses, it is necessary to provide their ON and OFF time as long as the interrupt time of the CPU module or longer.
The interrupt time of individual CPU module is shown below:

CPU module Type Name	Interrupt Time
High Performance model QCPU, Process CPU	1ms
QnACPU	5ms

(2) The set value cannot be changed while a count operation performed according to the UDCNT2 instruction is being executed (while the command input is ON).To change the set value, first turn the command input off.

(3) Counters designated by the UDCNT2 instruction cannot be used by any other instruction. If they are used by other instructions, they will not be capable of returning an accurate count.

(4) The UDCNT2 instruction can be used as many as 5 times within all the programs being executed.
The sixth and the subsequent UDCNT2 instructions are not processed.

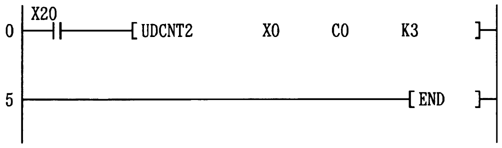
[Operation Errors]

(1) There are no operation errors associated with the UDCNT2 instruction.

[Program Example]

(1) The following program performs a count operation as instructed by C0 (count up or down) on the status of X0 and X1 after X20 has gone ON.

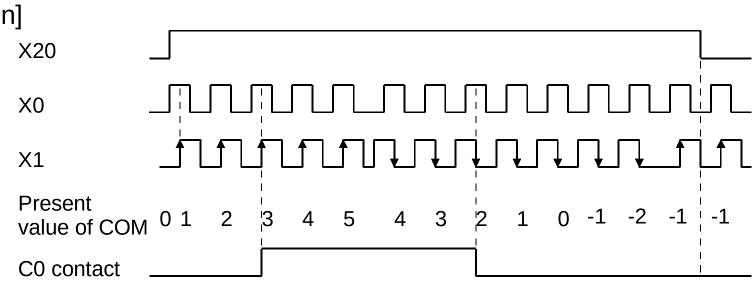
[Ladder Mode]



[List Mode]

Steps	Instruction	Device
0	LD	X20
1	UDCNT2	X0 C0 K3
5	END	

[Operation]

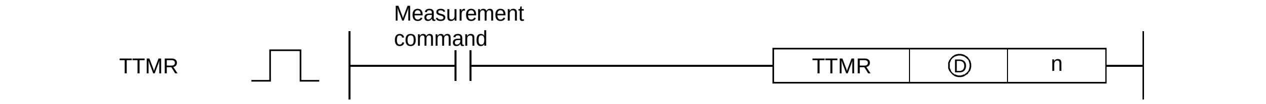


QCPU			QnA	Q4AR
PLC CPU		Process CPU		
Basic	High Performance			
×	○	○	○	○

6.8.3 Teaching timer (TTMR)

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant K, H	Other
	Bit	Word		Bit	Word				
Ⓓ	–	○		–					–
n	–	○		○					–

[Instruction Symbol] [Execution Condition]



[Set Data]

Set Data	Meaning	Data Type
Ⓓ	• Ⓓ +0: Storage device for measurement value • Ⓓ +1: For CPU module system use	BIN 16 bits
n	• Measurement value multiplier	

[Functions]

(1) The time that the measurement command is on is measured in units of seconds, then multiplied by the multiplier designated by n and the product is stored at the device designated by Ⓓ.

(2) When the measurement command goes from OFF to ON, the device designated by Ⓓ or Ⓓ +1 is cleared.

(3) The multipliers that can be designated by n are as shown below:

Ⓢ	Multiplier
0	1
1	10
2	100

POINTS

(1) Time measurements are conducted when the TTMR instruction is executed.
Using the JMP or similar instruction to jump the TTMR instruction will make it impossible to get an accurate measurement.

(2) Do not change the multiplier designated by n while the TTMR instruction is being executed.
Changing this multiplier will result in an inaccurate value being returned.

(3) The TTMR instruction can also be used in low speed type programs.

(4) The device designated by Ⓓ +1 is used by the CPU system, so users should not change its value.
If users do change this value, the value stored in the device designated by Ⓓ will no longer be accurate.

(4) No processing is performed when the value specified by "n" is other than 0 to 2.

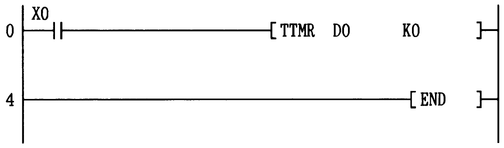
[Operation Errors]

(1) There are no errors associated with the TTMR instruction.

[Program example]

(1) The following program stores the amount of time that X0 is ON at D0.

[Ladder Mode]



[List Mode]

Steps	Instruction	Device
0	LD	X0
1	TTMR	D0
		K0
4	END	

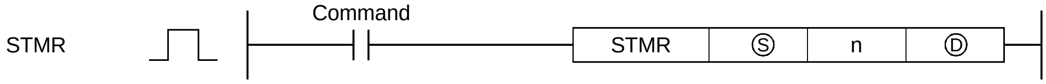
QCPU			QnA	Q4AR
PLC CPU		Process CPU		
Basic	High Performance			
×	○	○	○	○

6.8.4 Special function timer (STMR)

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant K, H	Other
	Bit	Word		Bit	Word				
Ⓢ	–	△ ∗	–	–					–
n	○	–	–	–					–
Ⓓ	–	○	○	○					–

∗: Can be used only by timer (T) data

[Instruction Symbol] [Execution Condition]



[Set Data]

Set Data	Meaning	Data Type
Ⓢ	Timer number	Word
n	Set value	BIN 16 bits
Ⓓ	• Ⓓ +0: Off delay timer output • Ⓓ +1: One shot timer output after OFF • Ⓓ +2: One shot timer output after ON • Ⓓ +3: ON delay timer output	Bit

[Functions]

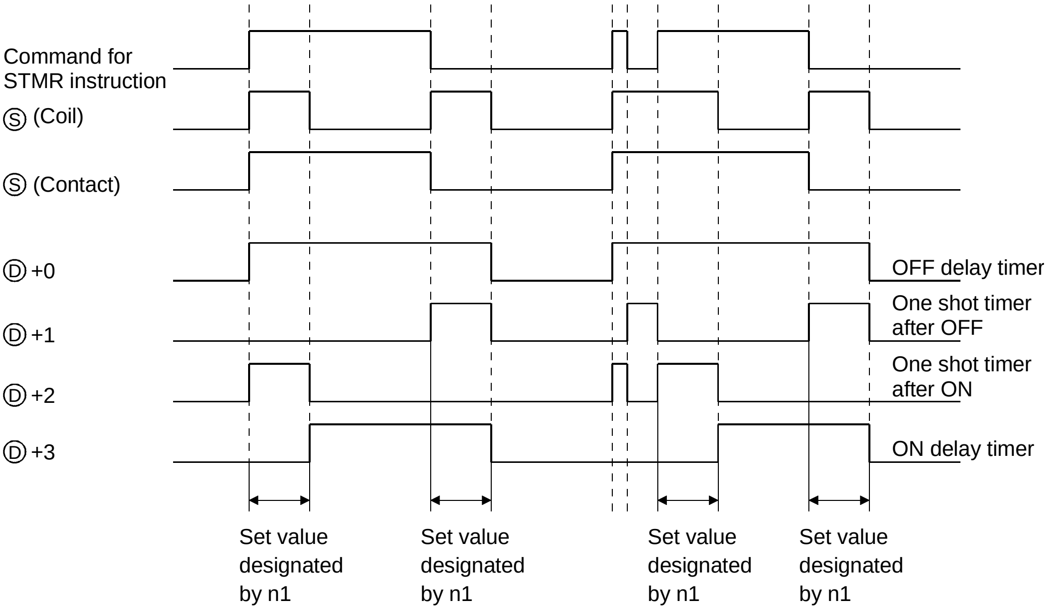
(1)The STMR instruction uses the 4 points from the device designated by Ⓓ to perform four types of timer output.

- OFF delay timer output (Ⓓ +0)
  Goes ON at the leading edge of the command for the STMR instruction, and, after the trailing edge of the command, goes OFF when the amount of time designated by n has passed.
- One shot timer output after OFF (Ⓓ +1)
  Goes ON at the trailing edge of the command for the STMR instruction, and goes OFF when the amount of time designated by n has passed.
- One shot timer output after ON (Ⓓ +2)
  Goes ON at the leading edge of the command for the STMR instruction, and goes OFF either when the amount of time designated by n has passed, or when the command for the STMR instruction goes OFF.
- ON delay timer output (Ⓓ +3)
  Goes ON at the trailing edge of the timer coil, and after the trailing edge of the command for the SRMR instruction, goes OFF when the amount of time designated by n has passed.

(2) The timer coil designated by Ⓢ goes ON at the leading edge of the command for the STMR instruction, and begins the measurement of the present value.

- The timer coil measures to the point where the value reaches the set value designated by n, then enters a time up state and goes OFF.
- If the command for the SRMR instruction goes OFF before the timer coil reaches the time up state, it will remain ON.
  Timer measurement is suspended at this time.
  When the STRM instruction command goes ON once again, the present value will be cleared to 0 and measurement will begin once again.

(3) The timer contact goes ON at the leading edge of the command for the STMR instruction, and after the trailing edge is reached, the timer coil goes OFF at the trailing edge of the STMR instruction command.
The timer contact is used by the CPU module system, and cannot be used by the user.


Command for STMR instruction
Ⓢ (Coil)
Ⓢ (Contact)
Ⓓ+0
Ⓓ+1
Ⓓ+2
Ⓓ+3
OFF delay timer
One shot timer after OFF
One shot timer after ON
ON delay timer
Set value designated by n1
Set value designated by n1
Set value designated by n1
Set value designated by n1

(4) Measurement of the present value of the timer designated by the STMR instruction is conducted during the execution of the STMR instruction.
If the STMR instruction is jumped with the JMP or similar instruction, it will not be possible to get accurate measurement.

(5) Measurement unit for the timer designated by Ⓓ is identical to the low speed timer.

(6) A value between 1 to 32767 can be set for n.

(7) The timer designated by Ⓢ cannot be used by the OUT instruction.
If the STMR instruction and the OUT instruction use the same timer number, accurate operation will not be conducted.

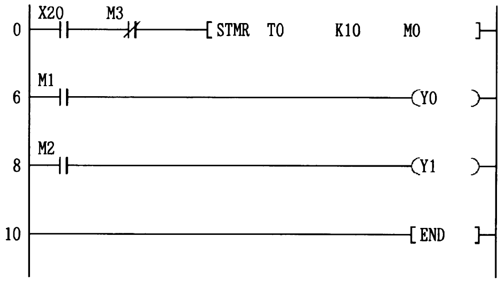
[Operation Errors]

(1) There are no errors associated with the STMR instruction.

[Program Example]

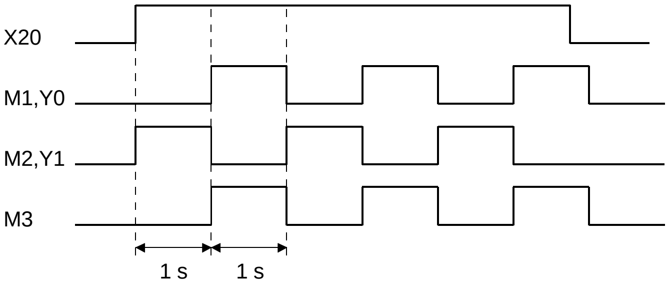
(1) The following program turns Y0 and Y1 ON and OFF once each second (flicker) when X20 is ON. (Uses the 100ms timer)

[Ladder Mode]



[List Mode]

Steps	Instruction	Device
0	LD	X20
1	ANI	M3
2	STMR	T0
		K10
		M0
6	LD	M1
7	OUT	Y0
8	LD	M2
9	OUT	Y1
10	END	

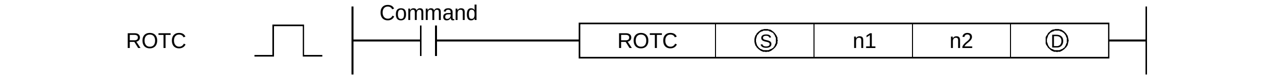


QCPU				QnA	Q4AR
PLC CPU		Process CPU			
Basic	High Performance				
×	○	○	○	○	○

6.8.5 Rotary table near path rotation control (ROTC)

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant	Other
	Bit	Word		Bit	Word				
Ⓢ	–	○		–					–
n	○	○		○					–
n2	○	○		○					–
Ⓓ	–	–		–					–

[Instruction Symbol] [Execution Condition]



[Set Data]

Set Data	Meaning	Data Type
Ⓢ	• Ⓢ +0: Measures table rpm (for system use)	BIN 16 bits
	• Ⓢ +1: Call station number	
	• Ⓢ +2: Call item number	
n1	• Number of divisions on table (from 2 to 32767)	
n2	• Number of low speed sections (value from 0 to n1)	
Ⓓ	• Ⓓ +0: A phase input signal	Bit
	• Ⓓ +1: B phase input signal	
	• Ⓓ +2: 0 point detection input signal	
	• Ⓓ +3: High speed forward rotation output signal (for system use)	
	• Ⓓ +4: Low speed forward rotation output signal (for system use)	
	• Ⓓ +5: Stop output signal (for system use)	
	• Ⓓ +6: High speed reverse rotation output signal (for system use)	
	• Ⓓ +7: Low speed reverse rotation output signal (for system use)	

[Functions]

(1) This control functions to enable near path rotation of the rotary table to the position of the station number designated by Ⓢ +1 in order to remove or deposit an item whose number has been designated by Ⓢ +2 on a rotary table with equal divisions of the value designated by n1.

(2) The item number and station number are controlled as items allocated by counterclockwise rotation.

(3) The system uses Ⓢ +0 as a counter to instruct it as to what item is at which number counting from station number 0.
Do not rewrite the sequence program data.
Accurate controls will not be possible in cases where users have rewritten the data.

(4) The value of n2 should be less than the number of table divisions that were designated by n1.

(5) Ⓓ +0 and Ⓓ +1 are A and B phase input signals that are used to detect whether the direction of the rotary table rotation is forward or reverse.

The direction of rotation is judged by whether the B phase pulse is at its leading or trailing edge when the A phase pulse is ON:

- When the B phase is at the leading edge : Forward rotation (clockwise rotation)
- When the B phase is at the trailing edge: Reverse rotation (counterclockwise rotation)

(6) Ⓓ +2 is the 0 point detection output signal that goes ON when item number 0 has arrived at the No. 0 station.
When the device designated by Ⓓ +2 goes ON while the ROTC instruction is being executed, Ⓢ +0 is cleared.
It is best to perform this clear operation first, then to begin near path rotation with the ROTC instruction.

(7) The data from Ⓓ +3 to Ⓓ +7 consists of output signals needed to control the table's operation.
The output signal of one of the devices from Ⓓ +3 to Ⓓ +7 will go ON in response to the execution results of the ROTC instruction.

(8) If operation results immediately prior to the ROTC instruction are OFF, all signals from Ⓓ +3 to Ⓓ +7 will be OFF without near path rotation controls having been performed.

(9) The ROTC instruction can be used only one time in all programs where it is executed.
Attempts to use it more than one time will result in inaccurate operations.

(10) No processing is performed when the value of Ⓢ+0 to Ⓢ+2, or the value of n2 is greater than n1.

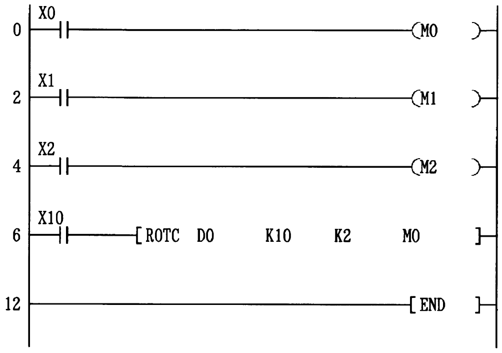
[Operation Errors]

(1) There are no errors associated with the ROTC instruction.

[Program Example]

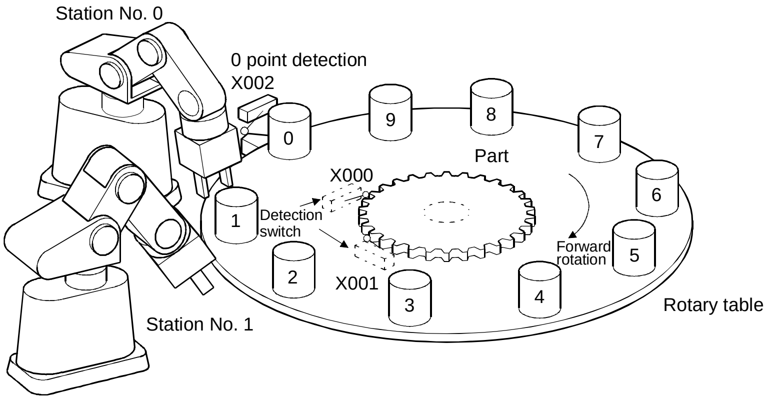
(1) The following program deposits the item at section D2 on a 10-division rotary table at the station at section D1, and the two sections ahead and behind this determine the rotation direction and control speed of the motor when the table is being rotated at low speed.

[Ladder Mode]



[List Mode]

Steps	Instruction	Device
0	LD	X0
1	OUT	M0
2	LD	X1
3	OUT	M1
4	LD	X2
5	OUT	M2
6	LD	X10
7	ROTC	D0 K10 K2 M0
12	END	

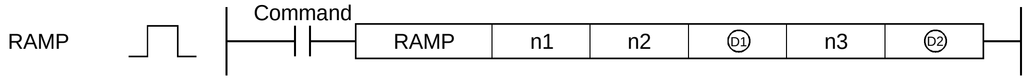


QCPU			QnA	Q4AR
PLC CPU		Process CPU		
Basic	High Performance			
×	○	○	○	○

6.8.6 Ramp signal (RAMP)

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant K, H	Other
	Bit	Word		Bit	Word				
n1	○	○						○	–
n2	○	○						○	–
D1	○	○						–	–
n3	○	○						○	–
D2	○	–						–	–

[Instruction Symbol] [Execution Condition]



[Set Data]

Set Data	Meaning	Data Type
n1	• Initial value	BIN 16 bits
n2	• Final value	
D1	• D1 +0: Present value • D1 +1: Number of times executed (for system use)	
n3	• Number of times moved	
D2	• D2 +0: Completion device • D2 +1: Selected bit where data is to be saved at completion.	Bit

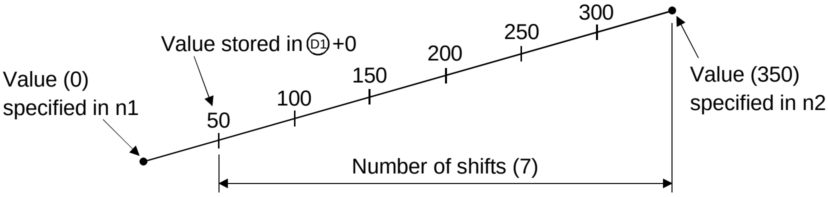
[Functions]

(1) Stores in D1+1 the value which varies the value specified in n1 to the value specified in n2 linearly by the number of shifts specified in n3.

The value to be stored in D1+1 is calculated every scan with the following expression.

$$\underbrace{\frac{\{(\text{Value specified in n2}) - (\text{Value specified in n1})\}}{(\text{Number of shifts})}}_{\text{One variation}} \times (\text{Number of executions})$$

0 is varied to 350 in six scans as shown below.



When the calculated one variation is indivisible, compensation is made to achieve the value specified in n2 by the number of shifts specified in n3.

Hence, a linear ramp may not be made.

(2) For n3, designate the number of scans required to move data from n1 to n2.
No processing is performed when n3 = 0.

(3) The system uses (D1) +1 to store the number of times the instruction has been executed.

(4) When the move is completed to the final value, the completion device designated by (D2) +0 will go ON.
The ON/OFF status of the completion device and the contents of (D1) +0 are determined by the ON/OFF status of the device designated by (D2) +1.

- When (D2) +1 is OFF, (D2) +0 will go OFF at the next scan, and the RAMP instruction will begin a new move operation from the value currently at (D1) +0.
- When (D2) +1 is ON, (D2) +0 will remain ON, and the contents of (D1) +0 will not change.

(5) When the command is turned OFF during the execution of this instruction, the contents of (D1) +0 will not change following this.
When the command goes ON again, the RAMP instruction will begin a new move from the present value at (D1) +0.

(6) Do not change the specified values in n1 and n2 before the completion device specified in (D2)+0 turns ON.
Since the same expression is used every scan to calculate the value stored in (D1)+1, changing n1/n2 may cause a sudden variation.

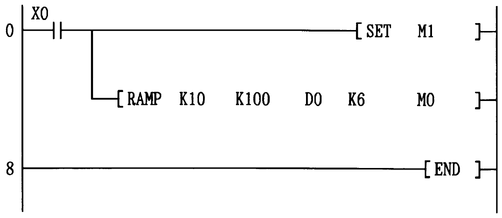
[Operation Errors]

(1) There are no operation errors associated with the RAMP instruction.

[Program Example]

(1) The following program changes the contents of D0 from 10 to 100 in a total of 6 scans, and saves the contents of D0 when the move has been completed.

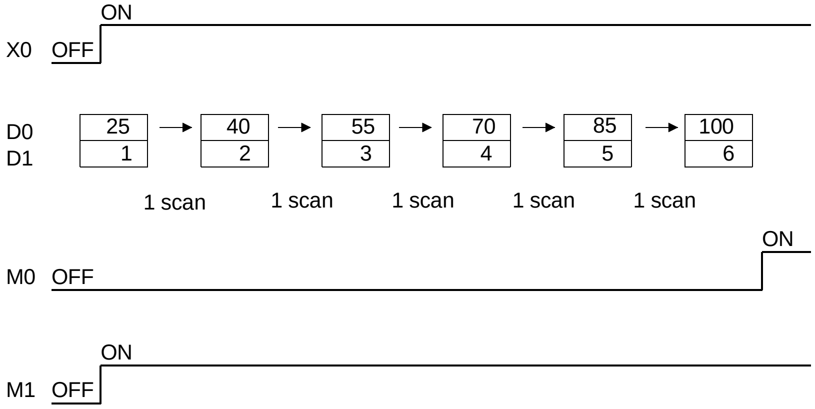
[Ladder Mode]



[List Mode]

Steps	Instruction	Device
0	LD	X0
1	SET	M1
2	RAMP	K10 K100 D0 K6 M0
8	END	

[Operation]



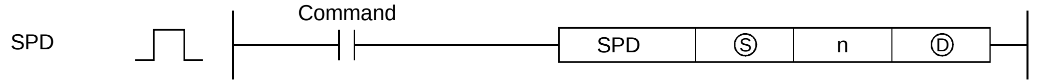
QCPU			QnA	Q4AR
PLC CPU		Process CPU		
Basic	High Performance			
×	○	○	○	○

6.8.7 Pulse density measurement (SPD)

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant	Other
	Bit	Word		Bit	Word				
Ⓢ	○ (X only)	—		○					—
n	△ *	△ *		—					—
Ⓓ	—	△ *		○					—

*: Local devices and the file registers set for individual programs cannot be used.

[Instruction Symbol] [Execution Condition]

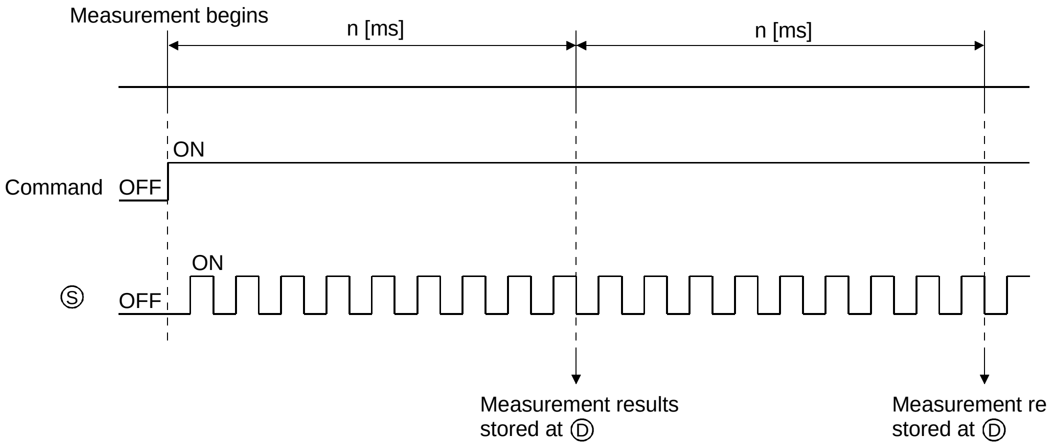


[Set Data]

Set Data	Meaning	Data Type
Ⓢ	• Pulse input	Bit
n	• Measurement time (unit: ms)	BIN 16 bits
Ⓓ	• Head number of device which stores measurement results	

[Functions]

(1) Input from the device designated by Ⓢ is counted for just the amount of time designated by n1, and results of the count are stored in the device designated by Ⓓ.


Measurement begins
n [ms]
n [ms]
ON
Command
OFF
ON
Ⓢ
OFF
Measurement results stored at Ⓓ
Measurement results stored at Ⓓ

(2) When measurement directed by the SPD instruction has been completed, measurement is done again from 0.
To suspend measurement directed by the SPD instruction, turn the command OFF.

POINTS

(1) The SPD instruction registers the data from the argument device in the CPU module work area, and the actual count operation is conducted during a system interrupt.
(The device data registered to the work area of the CPU module are cleared when the command input is turned OFF or when the CPU module is STOPped and then RUN.)
Therefore, to count the pulses, it is necessary to provide their ON and OFF time as long as the interrupt time of the CPU module or longer.
The interrupt time of individual CPU module is shown below:

CPU module Type Name	Interrupt Time
High Performance model QCPU, Process CPU	1ms
QnACPU	5ms

(2) • When QCPU is used:
The instruction is not processed when n=0.
• When QnACPU is used:
The instruction is not processed when n=0 or when n is not a multiple of 5.

(3) The SPD instruction can be used as many as 6 times within all the programs being executed.
The seventh and the subsequent SPD instructions are not processed.

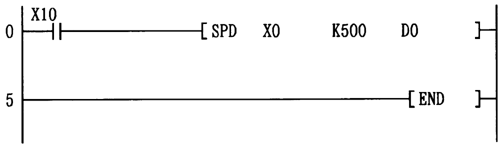
[Operation Errors]

(1) There are no operation errors associated with the SPD instruction.

[Program Example]

(1) The following program measures the pulses input to X0 for a period of 500 ms when X10 goes ON, and stores the result at D0.

[Ladder Mode]



[List Mode]

Steps	Instruction	Device
0	LD	X10
1	SPD	X0 K500 D0
5	END	

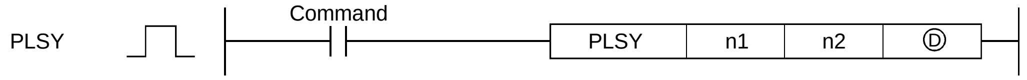
QCPU			QnA	Q4AR
PLC CPU		Process CPU		
Basic	High Performance			
×	○	○	○	○

6.8.8 Fixed cycle pulse output (PLSY)

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant	Other
	Bit	Word		Bit	Word				
n1	○	○							–
n2	○	○							–
Ⓓ	△ ∗	–							–

∗: Only output (Y) can be used

[Instruction Symbol] [Execution Condition]



[Set Data]

Set Data	Meaning	Data Type
n1	• Number of device where frequency is set	BIN 16 bits
n2	• Device No. of device that sets the number of outputs	
Ⓓ	• Number of device where pulse output is conducted	Bit

[Functions]

(1) Outputs a pulse at a frequency designated by n1 the number of times designated by n2, to the output module with the output signal (Y) designated by D1.

(2) Frequencies between 1 to 100 Hz can be designated by n1.
If n1 is other than 1 to 100 Hz, the PLSY instruction will not be executed.

(3) The number of outputs that can be designated by n2 is between 1 to 65535 (0000$_H$ to 0FFFF$_H$).

(4) Only an output number corresponding to the output module can designated for pulse output at Ⓓ.

(5) Pulse output commences with the command leading edge of the PLSY instruction.
Do not turn the command of the PLSY instruction OFF during pulse output.
Pulse output is suspended when the PLSY instruction command goes OFF.

POINT

(1) The PLSY instruction registers the argument device data in the CPU module work area, and the actual output operation is processed during system interrupts.
(The device data registered to the work area of the CPU module are cleared when the command input is turned OFF or when the CPU module is STOPped and then RUN.)
Therefore, to count the pulses, it is necessary to provide their ON and OFF time as long as the interrupt time of the CPU module or longer.
The interrupt time of individual CPU module is shown below:

CPU module Type Name	Interrupt Time
High Performance model QCPU, Process CPU	1ms
QnACPU	5ms

(2) Do not change the argument of the PLSY instruction during pulse output by the PLAY instruction (while the command input is ON).
Turn OFF the command input before changing the argument.

(3) For this reason, the PLSY instruction can be used only once in the entire program executed by the CPU module.

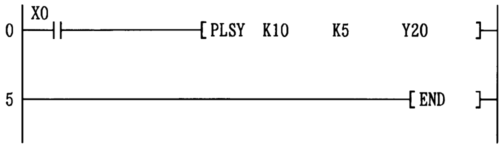
[Operation Errors]

(1) There are no operation errors associated with the PLSY instruction.

[Program Example]

(1) The following program outputs a 10 Hz pulse 5 times to Y20 when X0 is ON.

[Ladder Mode]



[List Mode]

Steps	Instruction	Device
0	LD	X0
1	PLSY	K10 K5 Y20
5	END	

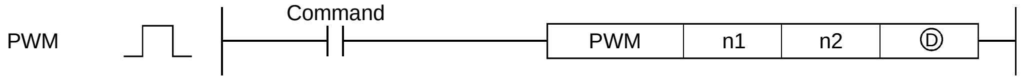
QCPU				
PLC CPU		Process CPU	QnA	Q4AR
Basic	High Performance			
×	○	○	○	○

6.8.9 Pulse width modulation (PWM)

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant	Other
	Bit	Word		Bit	Word				
n1	○	○							—
n2	○	○							—
Ⓓ	△ *	—							—

*: Only output (Y) can be used

[Instruction Symbol] [Execution Condition]

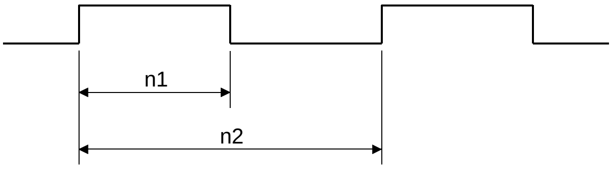


[Set Data]

Set Data	Meaning	Data Type
n1	• Number of device where ON time is set	BIN 16 bits
n2	• Number of device where cycle is set	
Ⓓ	• Number of device which will perform pulse output	Bit

[Functions]

(1) Outputs the pulse of the cycle set by n2, for the amount of time ON designated by n1, to the output module designated by Ⓓ .



(2) The setting ranges for n1 and n2 are shown below:

CPU module Type Name	Setting Range for n1* and n2 [ms]
High Performance model QCPU, Process CPU	1 to 65535 (0001H to 0FFFFH)
QnACPU	5 to 65535 (0005H to 0FFFFH)

* The value designated for n1 should be the same as the value designated for n2 or smaller.

[Operation Errors]

(1) There are no operation errors associated with the PWM instruction.

POINT

(1) The PWM instruction registers the designated device data to the work area of the CPU module.
The actual output operation is processed as the interruption by the CPU module.
(The device data registered to the work area of the CPU module is cleared when the command input is turned OFF or when the CPU module is STOPped and then RUN.)
The interrupt time of individual CPU module is shown below:

CPU module Type Name	Interrupt Time
High Performance model QCPU, Process CPU	1ms
QnACPU	5ms

For this reason, the PWM instruction can be used only once within all the programs being executed by the CPU module.

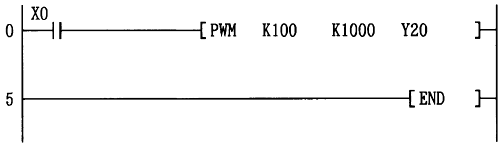
(2) The instruction is not processed in the following cases:
- When both n1 and n2 are 0
- When n1 and n2 are not multiples of 5 (only when QnACPU is used)
- When n2 ≥ n1

(3) Do not change the arguments of the PWM instruction while pulses are being output by the PWM instruction (while the command input is ON).
Before changing the arguments, turn OFF the command input.

[Program Example]

(1) The following program outputs a 100 ms pulse once each second to Y20 when X0 is ON.

[Ladder Mode]



[List Mode]

Steps	Instruction	Device
0	LD	X0
1	PWM	K100
		K1000
		Y20
5	END	

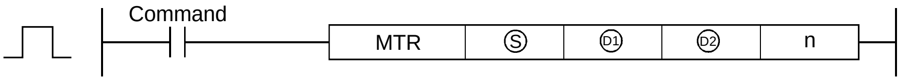
QCPU			QnA	Q4AR
PLC CPU		Process CPU		
Basic	High Performance			
×	○	○	○	○

6.8.10 Matrix input (MTR)

Set Data	Usable Devices								
	Internal Devices (System, User)		File Register	MELSECNET/10(H) Direct J□\□		Special Function Module U□\G□	Index Register Zn	Constant	Other
	Bit	Word		Bit	Word				
Ⓢ	○	—							—
Ⓓ1	○	—							—
Ⓓ2	○	—							—
n	○	○							—

[Instruction Symbol] [Execution Condition]

MTR



[Set Data]

Set Data	Meaning	Data Type
Ⓢ	• Head input device	Bit
Ⓓ1	• Head output device	
Ⓓ2	• Head number of device that will store matrix input data	
n	• Number of input rows	BIN 16 bits

[Functions]

(1) Successively reads the input from 16 points starting from the input number designated by Ⓢ, multiplied by n-rows, then stores the data fetched in this operation from the device designated by Ⓓ2 onward.

(2) One row (16 points) can be fetched in 1 scan.

(3) Fetching from the first to the nth row is progressively repeated.

(4) The first through the 16th points store the first row of data and the next 16 points store the second row of data at the devices following the device designated by Ⓓ2.
For this reason, the space of 16xn-points from the device designated by Ⓓ2 are occupied by the MTR instruction.

(5) Ⓓ1 is the output needed to select the row which will be fetched, and the system automatically turns it ON and OFF.
It uses the n-points from the device designated by Ⓓ1.

(6) Only device numbers divisible by 16 can be designated for Ⓢ, Ⓓ1 and Ⓓ2.

(7) The value for n2 is not between 2 to 8. (Error No. 4100)

(8) No processing is performed in the following cases.
- The device number designated by Ⓢ, Ⓓ1, or Ⓓ2 is not divisible by 16.
- The device designated by Ⓢ is outside the actual input range.
- The device designated by Ⓓ1 is outside the actual output range.
- The space 16 x n-points following the device designated by Ⓓ2 exceeds the relevant device range.
- The value for n2 is not between 2 and 8.

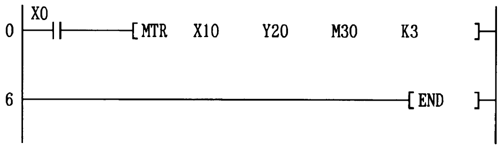
[Operation Errors]

(1) There are no errors associated with the MTR instruction.

[Program Example]

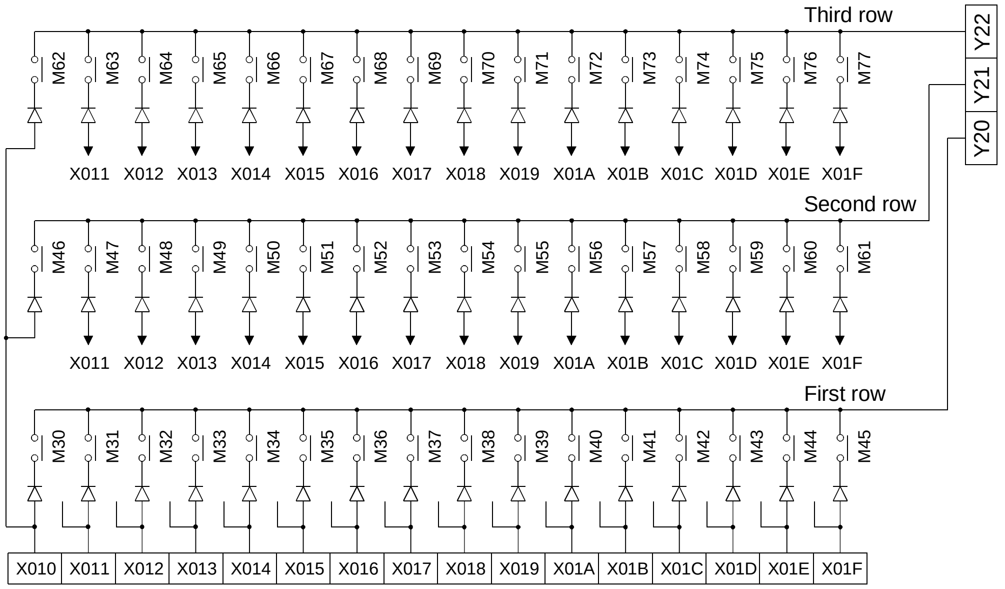
(1) The following program fetches 16 points x 3 rows starting from X10 when X0 is ON, and stores it from M30 onward.

[Ladder Mode]



[List Mode]

Steps	Instruction	Device
0	LD	X0
1	MTR	X10 Y20 M30 K3
6	END	


Third row
Y22
Y21
Y20
M62 M63 M64 M65 M66 M67 M68 M69 M70 M71 M72 M73 M74 M75 M76 M77
X011 X012 X013 X014 X015 X016 X017 X018 X019 X01A X01B X01C X01D X01E X01F
Second row
M46 M47 M48 M49 M50 M51 M52 M53 M54 M55 M56 M57 M58 M59 M60 M61
X011 X012 X013 X014 X015 X016 X017 X018 X019 X01A X01B X01C X01D X01E X01F
First row
M30 M31 M32 M33 M34 M35 M36 M37 M38 M39 M40 M41 M42 M43 M44 M45
X010 X011 X012 X013 X014 X015 X016 X017 X018 X019 X01A X01B X01C X01D X01E X01F

[Caution]

(1) Note that the MTR instruction directly operates on actual input and cutput.
The output Ⓓ1 that had been turned ON by the MTR instruction does not turn OFF when the MTR command turns OFF. Turn OFF the specified output Ⓓ1 in the sequence program.

(2) An MTR instruction execution interval must be longer than the total of response time of input and output modules.
If the set interval is shorter than the value indicated above, an input cannot be read correctly.
If the scan time in a sequence program is short, select the constant scan and set the scan time longer than the total of response time.